Universidade de Aveiro
2012
Departamento de Eletrónica, Telecomunicações e Informática

André Lourenço Tomás

Identificação de Utentes em Sessões Web com o Cartão de Cidadão

André Lourenço Tomás

Identificação de Utentes em Sessões Web com o Cartão de Cidadão

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Prof. Doutor André Ventura da Cruz Marnoto Zúquete, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e do Prof. Doutor Cláudio Teixeira, equiparado a investigador auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

O júri

Presidente	Professor Doutor Osvaldo Manuel da Rocha Pacheco, Professor Auxiliar, Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.
Vogal – Arguente Principal	Professor Doutor Carlos Nuno da Cruz Ribeiro, Professor Auxiliar, Departamento de Engenharia Informática do Instituto Superior Técnico da Universidade Técnica de Lisboa.
Vogal – Orientador	Professor Doutor André Ventura da Cruz Marnôto Zúquete, Professor Auxiliar, Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro
Vogal – Coorientador	Doutor Cláudio Jorge Vieira Teixeira, Equiparado a Investigador Auxiliar, Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Agradecimentos

Quero agradecer aos meus pais por toda a ajuda e o apoio que demonstraram ao longo do meu percurso académico e espero um dia poder retribuir o contributo que eles deram para o meu conhecimento.

Agradeço a disponibilidade do meu orientador Prof. Dr. André Zúquete e do coorientador Prof. Cláudio Teixeira que através de opiniões e ideias criativas sempre me motivaram e mostraram o seu apoio durante o desenvolvimento deste trabalho.

Quero também agradecer aos meus colegas de trabalho, nomeadamente ao Rui Ferreira, Daniel Corujo e ao Rodolphe Marques que ajudaram imenso na revisão desta tese e contribuíram com o seu tempo e conhecimento com o qual consegui agilizar algumas questões que foram aparecendo no decorrer do trabalho.

Não posso deixar também de agradecer a um amigo de longa data, Ruben Jorge, que se propôs desde cedo a me ajudar na realização deste trabalho.

palavras-chave

Single Sign On, Cartão de Cidadão, autenticação, autenticação mútua, sessões “web” seguras.

Resumo

Os diferentes métodos de autenticação existentes atualmente conferem diferentes níveis de segurança aos sistemas de autenticação. Métodos como credenciais guardadas em “smartcards” podem atingir um elevado nível de segurança. Por outro lado métodos baseados em palavra-chave têm um baixo nível de segurança uma vez que estas podem ser roubadas ou deduzidas por um atacante. No entanto, implementações deste tipo de autenticação são bastante mais comuns pela sua facilidade de implementação e utilização.

No âmbito deste trabalho pretende-se dar a conhecer alguns dos sistemas de autenticação mais comuns bem como a possibilidade da utilização do cartão de cidadão como mecanismo de autenticação. É pretendido também analisar os servidores Web com capacidade de autenticação mútua e respetiva capacidade de gestão de sessões SSL nos mesmos.

Keywords Single sing on. Web Authentication, Smartcard, HTTPS, secure web session, Portuguese citizen card.

Abstract The different currently existing authentication methods provide different levels of authentication security. Methods such as credentials stored on smartcards can achieve a high level of security. However, methods based on keywords have a low level of security, since these may be stolen or deducted by an attacker. Moreover, implementations of this type of authentication are much more common due to its ease of implementation and usage. This work aims to introduce some of the most common authentication systems, as well as using the citizen card as an authentication mechanism. It also intends to analyze the web server's capability for mutual authentication and their respective management capability of SSL sessions.

Conteúdo

Índice de Figuras

Indice de Tabelas

Dicionário de acrónimos

ARTIFACT	“Artifact Resolution Protocol” (Protocolo de resolução de artefactos). Fornece um mecanismo pelo qual as mensagens do protocolo SAML podem ser passadas por referência utilizando para isso um valor de tamanho fixo chamado de artefacto.
CC	Cartão de Cidadão
Cid	Identificação da sessão do cliente
FORM	No contexto HTML representa um formulário que é composto por controlos que podem ser “Botões”, “Caixas de Textos”, “Caixas de Senha”, entre outros.
GC	Google Chrome
HTML	HyperText Markup Language
HTTPS	Hypertext Transfer Protocol Secure
IdP	Identity Provider
IE	Internet Explorer
MDSSO	Multi-domain Single Sign-On
NAT	Network Address Translation
OASIS	Organization for the Advancement of Structured Information Standards
OCSP	Online Certificate Status Protocol
PKI	Public Key Infrastructure
PIN	Personal Identification Number
SAML	Security Assertion Markup Language
SOAP	Simple Object Access Protocol
SP	Service Provider
SSL	Secure Rockets Ayer
SSO	Single Sign-On
STS	Security Toquem Service
TLS	Transporte Ayer Security
UA	User Agent
URI	Unique Resource Identifier
URL	Uniform Resource Locator
Web SSO	Web Single Sign-On
WS-Federation	Web Services Federation Language
WS-Security	Web Services Security
XHTML	Extensible Hypertext Mark-up Language
XML	Extensible Markup Language

1 Introdução

A segurança online é um aspeto cada vez mais importante na atualidade. À medida que as tecnologias de segurança evoluem, é cada vez mais importante considerar que os utilizadores de serviços online pretendem ter sempre a maior segurança possível, no que diz respeito tanto à sua privacidade, como à das transações que efetuam. Para que isto seja possível é indispensável considerar a autenticação, visto que é durante este processo que os intervenientes ficam a conhecer-se, e a confiar um no outro, para que depois estes possam iniciar uma comunicação cifrada e, por conseguinte, tornar a mesma segura para todo o tipo de operações ou trocas de informação.

Não podemos esquecer também o facto de que, quando um utilizador acede a um conjunto de informação online, esta pode não estar totalmente centralizada. Em concreto, para que o utilizador obtenha informação ou realize operações, por vezes terá de aceder a vários serviços vindos de diferentes fornecedores de serviços, tendo para isso que passar pelo processo de autenticação em todos eles.

"Tokens" e "smartcards" começam a ser cada vez mais usados para garantir a autenticação e, seguidamente, a cifra da informação, tentando garantir uma maior segurança na autenticação e comunicação entre dois intervenientes.

Como parte do processo de informatização, o governo português aplicou em 2008/2009 uma reforma no processo de autenticação dos seus cidadãos, criando como mecanismo de autenticação para os mesmos o Cartão de Cidadão (CC). Este cartão tem a capacidade de fornecer mecanismos de autenticação e assinatura digital a cada individuo.

1.1 Enquadramento

O crescimento da utilização da Internet por parte da população mundial tem levado cada vez mais ao aparecimento de novas tecnologias web, que por sua vez, têm vindo a disponibilizar um crescimento de serviços para os utilizadores. Sabendo que existem vários tipos de informação na web, são criados serviços para fazer a gestão dessa informação da melhor forma possível. Alguns destes serviços são apenas de caráter informativo e possuem informação não crítica. Esta informação é normalmente sobre assuntos do mundo, como um jornal "online" ou informações acerca do estado do tempo numa determinada região. Para este tipo de serviços, a autenticação do utilizador não é um tema crítico sendo que, habitualmente, não é importante verificar as permissões de acesso de um utilizador, ou até mesmo se este pretende ter acesso autenticado. É

importante sim, neste caso, certificar que as noticias são de confiança e estão a ser visualizadas no sítio correto.

No entanto, cada vez mais os utilizadores utilizam serviços e processos "online" onde os seus dados pessoais são necessários. Certas empresas, nomeadamente bancos e outras instituições financeiras, dispõem de informações críticas sobre cada utilizador, tais como dados da sua conta bancária e os seus níveis de acesso.

Foram então criados serviços com o propósito de gerir este tipo de informação, nos quais é de extrema importância verificar que um utilizador seja quem diz ser, fazendo com que este apenas tenha acesso à informação depois de se identificar ou autenticar.

Para este último tipo de serviços, cada vez mais se tenta chegar a uma forma que se prove infalível de autenticar um utilizador sem que terceiros o consigam impersonificar, ou de algum modo violar o acesso a este tipo de informação sensível. Este problema fez com que várias arquiteturas e modelos de autenticação tenham vindo a ser desenvolvidos para impedir tentativas de obter informação à qual um utilizador não tenha permissões de acesso.

Visando colmatar o problema de acesso à informação online por parte de um utilizador, foram desenvolvidas algumas arquiteturas de autenticação. O exemplo mais utilizado actualmente é designado de SSO (Single Sign On), que permite ao utilizador uma única autenticação num serviço, conseguindo com a mesma autenticar-se e obter informações de muitos outros serviços.

1.2 Motivação

Sendo que a segurança da informação online é um motivo de extrema importância, como referido nas secções anteriores, é essencial explorar a capacidade de autenticar com segurança um utilizador e o serviço que este pretende utilizar. O objetivo é garantir a autenticidade de ambos numa comunicação, antes de se iniciar o processo de troca de informação entre eles.

Caso se verifique alguma anomalia ou a dificuldade de implementação deste tipo de autenticação devido a restrições por parte dos navegadores, ou por parte dos fornecedores de serviços existentes, é relevante projetar uma solução para os problemas encontrados.

O tema deste trabalho vem de encontro ao existente conjunto de limitações dos sistemas de autenticação utilizados hoje em dia, mais propriamente a autenticação com o cartão português de identidade, CC. Para além estar pouco divulgado e utilizado atualmente na web pelos cidadãos portugueses, apresenta limitações no uso do mesmo para autenticação online. Um outro fator ao qual se pretende dar algum enfase é ao número de serviços disponibilizados nos

dias de hoje que utiliza este sistema. Apesar de este cartão ter sido pensado para um crescente futuro tecnológico de identificação online, não está tão presente quanto seria esperado na Internet ao fim de 3 anos de existência.

Assim, esta dissertação propõe processos para a resolução dos problemas referidos anteriormente, assim como uma utilização prática do CC para autenticação na "web".

1.3 Objetivo

Tendo como base a segurança na autenticação online, este trabalho visa, em primeiro lugar, testar um sistema de autenticação online com base no CC, bem como o seu comportamento na utilização de sessões HTTPS (Hypertext Transfer Protocol Secure), com recurso a autenticação mútua, em que ambas as partes da comunicação se autenticam uma com a outra utilizando um túnel TLS (Transport Layer Security). Este comportamento é testado em diferentes navegadores de acesso à Internet, com o objetivo de garantir um elevado grau de compatibilidade com os mesmos, de modo a abranger a maioria dos internautas que utilizam serviços na web.

Numa segunda fase, e atendendo aos resultados obtidos na primeira fase, este trabalho visa a criação de um serviço de autenticação com capacidades de SSO com o objetivo de disponibilizar aos utilizadores, através de uma entidade única, acesso a diferentes fornecedores de identidades. Desta forma, é permitida a utilização do CC por um mesmo serviço sem que para isso seja necessário existir uma dependência do navegador de acesso a internet ou suporte direto dos fornecedores destes serviços.

1.4 Estrutura da dissertação

Este trabalho encontra-se estruturado da seguinte forma. A secção 2 providencia uma definição inicial das tecnologias existentes para a criação de ligações seguras, assim como os processos necessários à criação das mesmas, passando depois para as arquiteturas de autenticação federada, tentando demostrar a relação entre os protocolos e mecanismos de autenticação cumprindo assim o estado de arte.

Inicia-se na secção 3 o estudo do controlo de sessões HTTP e SSL, possibilitando assim à secção 4, dar a conhecer as ideias e tecnologias de suporte que levaram ao desenvolvimento da arquitetura apresentada neste trabalho. Na secção 5 é apresentada uma nova arquitetura de autenticação SSO. Na secção 6 é descrito o processo de implementação e passos necessários à utilização deste trabalho e são mostrados os resultados obtidos após a implementação.

Por fim a secção seguinte (7), contém informação relativa à avaliação do trabalho e uma conclusão com algumas notas para trabalho futuro.

2 Estado de arte

A fim de se verificar a tecnologia existente no âmbito deste trabalho, bem como a possibilidade da sua implementação e viabilidade tendo em conta os mecanismos de autenticação e as API's existentes nas diferentes linguagens de programação, foi feito um estudo intensivo sobre as mesmas.

O estudo que se segue incidiu fundamentalmente sobre arquiteturas de autenticação mútua, com recurso a mecanismos de chave público-privada. Para tal foram analisados, em mais detalhe, os componentes e arquitetura dos mecanismos de autenticação mútua, os principais sistemas de autenticação SSO e os problemas associados a sessões HTTP e sessões HTTPS.

2.1 Autenticação

2.1.1 Segredo partilhado

Em redes de telecomunicações é muito importante que os intervenientes de uma comunicação, sejam eles pessoas, computadores ou serviços, se conheçam. Em concreto, ambas as partes da comunicação apresentaram a sua identidade e a autenticação resultante tenha sido aceite mutuamente. Uma forma comum de responder a este problema é utilizando segredos partilhados, contendo, normalmente uma chave que ambos os intervenientes conhecem. No entanto, o segredo partilhado apresenta um grave problema: o mesmo segredo tem de ser conhecido por ambos os intervenientes e tem de ser usado, não só na autenticação, mas também na cifra das mensagens trocadas entre os mesmos. Este processo permite a um utilizador à escuta na rede com intenções maliciosas deduzir e obter o segredo perdendo-se a segurança na comunicação pois as mensagens trocadas poderão ser escutadas ou alteradas por terceiros [1]. Existem outras formas de obter o segredo de um utilizador, tais como vírus ou "spywares", que podem registar palavras-chave ou chaves assimétricas usadas pelo utilizador enviando as mesmas ao atacante.

2.1.2 Par de chaves

A fim de colmatar as falhas no mecanismo de segredo partilhado apresentadas em 2.1.1, foi desenvolvido um mecanismo de chaves público-privada, também conhecido como "keypair", em que cada interveniente numa conversação, ou troca de mensagens, possui duas chaves, uma chave pública e uma chave privada. Estas chaves são por norma valores arbitrários com tamanho mínimo de 128 bytes [6]. O tamanho destas chaves é relativo à segurança desejada e à capacidade de processamento dos dispositivos que terão de tratar do processo de cifra/decifra

das mensagens. É importante considerar que, devido à diferença na capacidade de processamento, uma chave gerada, por exemplo num dispositivo móvel, poderá não ser suficiente para garantir o nível de segurança necessário na comunicação.

A chave pública é a chave que, como o nome indica, é do conhecimento público; a chave privada por sua vez é apenas do conhecimento do seu titular, estando normalmente protegida por um segredo conhecido unicamente pelo mesmo. O funcionamento das chaves é de todo o mais importante na garantia de segurança pois estas permitem que a informação seja cifrada, viaje dessa mesma forma para que não possa ser acedida por terceiros e, ao chegar ao destino, seja decifrada e acedida apenas pelo seu destinatário.

Existem então duas grandes potencialidades na utilização de pares de chaves. Em primeiro lugar a autenticação de, por exemplo, uma mensagem ou um documento público, bastando para isso adicionar uma assinatura efetuada com a chave privada para que, com a correspondente chave pública, seja validada, fazendo assim com que seja certificada a autoria ou autenticidade do documento ou mensagem. Uma segunda possibilidade passa por cifrar uma mensagem ou documento com a chave pública do destinatário fazendo com que apenas esse possa ler o conteúdo da mensagem, uma vez que possui a única chave privada existente, mantendo assim a confidencialidade da mesma [2].

A utilização de pares de chaves requer a existência de um algoritmo de cifra, de forma a computar a informação que se pretende autenticar ou cifrar. Assim, em 1976 foi proposto por Diffie e Hellman um modelo criptográfico de chave pública em que cada utilizador possui um par de chaves (*P* e *K*) sendo *P* a sua chave pública e *K* a sua privada [3]. Estas chaves têm de obedecer a um conjunto de regras e estão relacionadas através de uma fórmula matemática da seguinte forma:

- Se *w* corresponder a um texto legível, e *K()* representar a função da chave *K*, então a transformação do texto *w* através da função *K()* pode ser presentado por *K(w) =z*, daqui vem que *P(z) =w* onde *P()* representa a função da chave *P*, ou seja, *K* é a chave inversa da chave *P* onde, *P(K(w)) =w*.
- O cálculo do par de chaves *(K, P)* deve ser relativamente fácil de computar.
- Deve ser computacionalmente difícil derivar *K* a partir do conhecimento de *P*.
- As funções *P()* e *K()* devem ser computacionalmente fáceis de calcular para quem tenha o conhecimento das chaves.
- Deve ser computacionalmente difícil calcular *K()* sem conhecer a chave *K*.

Todas estas regras levam a que cada utilizador possa calcular o seu par de chaves *(S, P)* no seu computador.

- A chave *K* deve ser guardada de forma segura no computador.
- A chave *P* é distribuída a todos de forma pública.

Designa-se de algoritmo o processo de tratamento da informação ao texto legível através da aplicação das funções referidas no processo anterior, até que se obtenha um conjunto de dados ilegíveis. A complexidade do algoritmo afeta a quantidade de poder computacional necessário ao processamento do mesmo, quanto mais complexo este for, mais tempo demora ou mais poder de processamento é necessário o realizar.

Seja *n* o comprimento de entrada para um algoritmo *A*. Um algoritmo *A* é de tempo polinomial se a função *f(n)* do tempo de execução no pior caso de *A* é tal que *f(n) = O (n)* para uma constante *k*. Um algoritmo *A* é de tempo exponencial se não existe constante *k* tal que *f(n) = O (n)*. Algoritmos de tempo polinomial são computacionalmente eficientes (fáceis), e os algoritmos de tempo exponencial são computacionalmente ineficientes (difíceis). Diz-se que um problema é computacionalmente difícil se não se conhece qualquer algoritmo de tempo polinomial para resolvê-lo.

2.1.3 Certificados de chave pública

As chaves públicas podem ser guardadas num objeto designado por certificado de chave pública. Este contém, além da chave, o domínio a que pertence, o nome da entidade, a entidade emissora, a data de emissão, a validade do mesmo e, entre outros campos a cadeia de certificação desse certificado. Esta permite a qualquer instante validar o certificado como tendo sendo emitido por uma entidade certificada para o efeito, tornando-o assim um certificado de confiança.

Para que um certificado seja considerado válido, tem de obedecer a um conjunto de regras:

I. Possuir uma entidade emissora válida. Neste contexto, tem de ser uma entidade certificadora ou certificada por uma.
II. A data de validade do certificado tem de ser válida na data corrente (em que o certificado está a ser usado)
III. Não se pode encontrar na lista de certificados revogados.
IV. Nenhum dos certificados na sua cadeia de certificação pode ser inválido.

Para que um serviço seja de confiança, todos os anteriores fatores têm de ser válidos e o domínio onde o serviço se encontra registado tem de ser o mesmo presente no certificado. Um

objeto contendo estas características pode ser representado em criptografia pelo padrão X.509 [23] criado para dar suporte à PKI (Public Key Infrastucture) [12], esta estrutura é utilizada universalmente para autenticação de utilizadores ou serviços através de certificados de chave pública. A utilização destes certificados em conjunto com um canal de comunicação seguro, podem garantir autenticidade e confidencialidade suficiente à utilização de serviços críticos como "e-banking".

Quando um utilizador pretende aceder a um recurso ou serviço em que a importância da informação do mesmo é elevada ou mesmo crítica, (como é o caso da utilização de serviços "e-banking") é importante para o utilizador ter a certeza que o serviço ao qual está a aceder é efetivamente o serviço que pretende e está a ser fornecido pela entidade correta. Para que isto seja possível, a entidade de e-banking tem de ter um certificado válido com raiz numa entidade de confiança que entrega ao utilizador. Este por sua vez, efetua a validação do certificado enviado pelo serviço, através do protocolo Online Certificate Status Protocol (OCSP), e, sendo este válido, inicia uma comunicação segura com o serviço, cifrando as suas mensagens com a chave pública do mesmo, para que apenas o este possa aceder ao seu conteúdo [9].

Neste processo, o utilizador autenticou o serviço ao qual pretendia aceder e, como tal, tem a certeza de estar a comunicar com o serviço correto. É também depois deste processo que é estabelecida uma sessão segura, utilizando para isso um protocolo seguro, normalmente SSL/TLS, para que a comunicação seja cifrada e, por conseguinte, ter uma maior segurança na transmissão de dados.

Estes certificados podem ser guardados informaticamente em qualquer dispositivo de armazenamento de dados e enviados por qualquer meio, e.g. disco externo, memória flash, correio eletrónico, entre outros, não esquecendo a necessidade de guardar a chave privada associada a este certificado. A fim de garantir uma maior segurança da chave privada, uma vez que esta é necessária à utilização do certificado de chave pública, tanto essa chave como o certificado têm vindo a ser guardados em "tokens" de hardware, criados com o objetivo de divulgar o certificado de chave pública e proteger a chave privada.

2.1.4 Cartão de Cidadão

O CC é um documento de cidadania portuguesa. Como documento físico, permite a um cidadão português identificar-se presencialmente de forma segura. Como documento tecnológico, permite ao seu portador identificar-se perante serviços informatizados e autenticar documentos eletrónicos, juntando num só, as chaves indispensáveis ao relacionamento rápido e eficaz dos cidadãos com diferentes serviços públicos.

Este cartão contém um certificado de chave-pública e a correspondente chave privada, pertencendo à categoria de "smartcards".

A ideia base do "smartcard" surgiu pela primeira vez em 1968 por Helmut Gröttrup e Jürgen Dethloff que criaram um chip pequeno o suficiente para caber no bolso. Mais tarde, em 1977 foi adaptado num chip embutido num cartão plástico, por Michel Ugon. Desde este momento houve uma evolução deste tipo de cartões, sendo agora classificados em quatro categorias:

- Cartões de banda magnética.
- Cartões óticos.
- Cartões de memória.
- Cartões com microprocessador

O CC está inserido na categoria de Cartões com micro processador, pelo que estes serão alvo de uma explicação mais detalhada. Este tipo de cartão tem algumas capacidades de processamento e memória embutida pelo que permitem conter uma estrutura PKI [13].

É designado por PKI um conjunto de hardware, software, politicas e procedimentos necessários à criação, distribuição, uso, armazenamento e revogação de certificados digitais. Desta forma é possível mapear chaves públicas em entidades, armazena-las com segurança num repositório comum e revogá-las se necessário.

Com esta estrutura, PKI, é possível oferecer identificação, autenticação, armazenamento de dados e processamento [12].

Numa vertente digital, na qual este trabalho se insere, o CC promove o desenvolvimento das transações eletrónicas e navegação segura, fornecendo um sistema de autenticação seguro através da utilização de PKI [10]. Com este, são atualmente disponibilizados alguns serviços pelo Estado Português para autenticação em portais como, Finanças e Segurança Social. Uma das finalidades será agilizar a autenticação dos utilizadores evitando assim o nome de utilizador e palavra-chave muito esquecidos pelos utilizadores que não utilizam estes serviços frequentemente e fornecer um maior nível de segurança sendo que nestes serviços é aplicado o princípio da autenticação mútua, secção 2.1.5.

2.1.5 Autenticação mútua

Para que exista autenticação mútua é necessário estabelecer uma relação de confiança entre todos os intervenientes de uma comunicação. Neste género de autenticação, o processo é semelhante ao apresentado na secção 2.1.3 com a diferença de que o utilizador do serviço tem

também de apresentar um certificado válido, para que o serviço possa reconhecer e validar o utilizador do mesmo. Desta forma, ambos os intervenientes têm uma maior segurança relativamente a com quem estão a trocar mensagens, pois não é apenas o utilizador a autenticar o serviço, mas sim também o serviço a autenticar o utilizador, garantindo assim que na troca de mensagens apenas o seu destinatário pode aceder aos dados.

Para esta referida troca de mensagens em autenticação mútua se tornar o mais segura possível, existe um protocolo específico SSL (TLS a partir da versão 3 do SSL) [16], criado para dar suporte a este tipo de autenticação.

Existem vários protocolos de criptografia cujo objetivo é dar suporte a autenticação mútua sendo que os mais utilizados desde o seu aparecimento são a atualização do anterior, começando por SSL v1 passando pela v2 e v3 até finalmente se chegar ao TLS v1, uma vez que cada versão vem reparar falhas na versão anterior irá apenas ser apresentado o funcionamento da sua ultima versão (TLS v1).

O protocolo TLS permite que aplicações cliente-servidor possam comunicar através de uma rede de forma a prevenir “eavesdropping” [15], ou seja, previne que a informação que está a ser trocada seja escutada e compreendida por terceiros. Dado que a maioria dos protocolos de comunicação (HTTP por exemplo) pode ser utilizado com ou sem TLS, é necessário indicar ao servidor se o cliente vai iniciar uma ligação TLS ou não, existindo para esse efeito duas formas de o conseguir. No caso de uma sessão HTTP (HTTPS depois da ligação TLS), a primeira forma de notificar o servidor passa por utilizar um porto diferente, 433 em vez do regular 80. A outra forma de notificar o servidor passa por utilizar um mecanismo existente no TLS, por exemplo STARTTLS [17].

Depois desta decisão do uso de TLS, o cliente e servidor começam então por negociar uma ligação com informação de estado (stateful) utilizando um processo de “handshaking”. Durante este processo cliente e o servidor trocam vários parâmetros utilizados para estabelecer a segurança da ligação, esta troca de parâmetros está demonstrada na Figura 1.

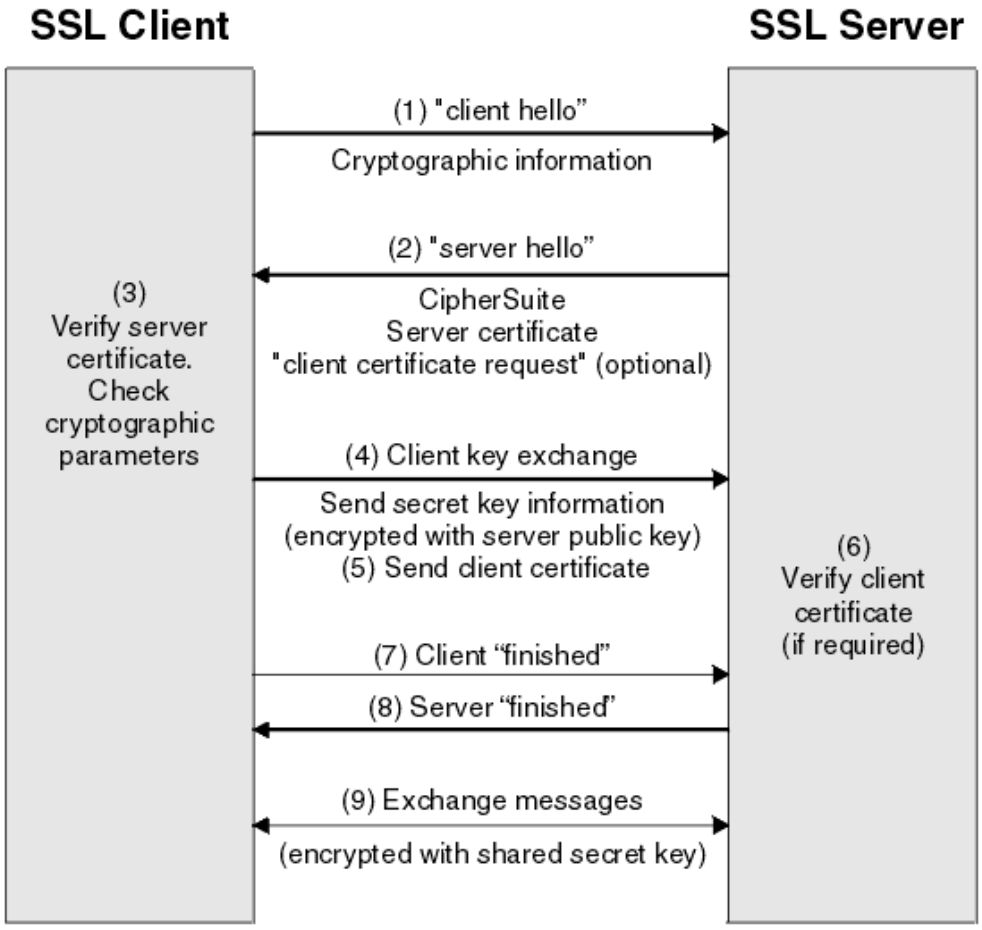

Figura 1 - Troca de parâmetros para início de uma sessão TLS

A troca de parâmetros acontece da seguinte forma:

1. O cliente envia ao servidor a versão de SSL suportada pelo mesmo, as definições de cifra, dados relativos à sessão e outras informações necessárias a uma sessão SSL.
2. O servidor responde ao cliente com os parâmetros anteriores, enviando também o seu certificado de chave pública. Se o cliente estiver a tentar aceder a um recurso que exija autenticação mútua, o servidor pede também o certificado de chave pública do cliente.
3. O cliente utiliza a informação recebida do servidor para autenticar o mesmo. Se o servidor não puder ser autenticado (por certificado inválido ou algum dos parâmetros não for suportado pelo cliente) o cliente reporta o problema e indica que uma ligação autenticada e cifrada não pode ser estabelecida. Se o servidor poder ser autenticado o cliente prossegue para o passo seguinte.
4. Utilizando toda a informação trocada no "handshake" até aqui, o cliente gera uma pré chave mestra da sessão cifrando a mesma com o a chave pública presente no certificado recebido no ponto 2, enviando de seguida esta chave cifrada para o servidor.

5. Se o servidor exigir autenticação do cliente, o cliente assina também um outro conjunto de informação única para este "handshake" e conhecida pelas duas partes, cliente e servidor. Neste caso, o cliente envia então a informação assinada e o seu certificado de chave pública no mesmo momento em que envia a pré chave mestre cifrada.
6. Se o servidor exigir autenticação do cliente, o servidor tenta autenticar o mesmo com a informação fornecida por este. Se o cliente não conseguir ser autenticado, a sessão é terminada. Se o cliente for autenticado com sucesso, o servidor usa a sua chave privada para decifrar a pré chave mestra enviada pelo cliente, executando de seguida os procedimentos necessários (também executados pelo cliente começando pela pré chave secreta) para gerar uma chave mestra.
7. Neste momento tanto o cliente como o servidor utilizam a chave mestra para gerar as chaves de sessão, sendo estas chaves simétricas utilizadas para cifrar e decifrar informação durante a sessão SSL/TLS, sendo assim possível verificar a integridade da mesma.
8. O cliente envia então uma mensagem para o servidor com o objetivo de o informar que mensagens futuras do cliente serão cifradas com a chave de sessão. Envia de seguida uma outra mensagem cifrada com informação de término do "handshake" da parte do cliente.
9. O Servidor de seguida envia uma mensagem ao cliente com a mesma informação de término de "handshake".

O processo de "handshake" está neste momento terminado, estando assim iniciada a sessão segura ao que o cliente e servidor utilizam as chaves de sessão para trocar informação entre si, podendo a qualquer momento renegociar esta sessão, repetindo os passos de 1 a 9. Se algum dos passos anteriores falhar, o "handshake" TLS falha e a ligação segura não é criada.

Apesar de que com todo este processo se conseguir uma ligação segura entre um cliente e um servidor, há outros problemas na autenticação e na gestão de identidades tanto dos clientes como dos servidores ou serviços, podemos considerar que, se o cliente pretender autenticar-se num servidor e depois utilizar um serviço fora do âmbito desse servidor, este esquema de autenticação não trás a resposta a este problema. Deste forma o cliente apenas poderá aceder à informação dentro do servidor no qual se autenticou.

2.1.6 Problema de autenticação e gestão de identidades online

Com base no problema referido no final da secção 2.1.5, pode pensar-se que a gestão de identidade e autenticação é um problema de segurança importante a resolver, tendo em conta o princípio de utilização de serviços com autenticação externa ao mesmo. Este problema levanta duas questões fundamentais à segurança numa comunicação:

a) Como podemos estabelecer e, em seguida, confirmar a identidade de um utilizador ou uma aplicação / sistema?
b) Como vamos informar todos os interessados que a identidade está estabelecida e que pode ser confiável?

Baseado numa situação onde o utilizador tem de se autenticar num servidor de autenticação, após o que o mesmo quer aceder a uma aplicação ou serviço num servidor diferente ou fornecedor de serviços (SP) [7], é preciso ter em atenção dois aspetos antes de pensar numa solução para o problema de a) e b) sendo eles:

I. **A autenticação do utilizador.** Esta componente diz respeito ao problema da gestão de identidade de um determinado utilizador. Ou seja, são identificados os intervenientes e determina-se o nível de confiança. Normalmente é feito por meio de autenticação fornecendo um nome de utilizador e respetiva palavra-chave, mas vários outros esquemas de autenticação de utilizador podem ser utilizados, tais como tokens de hardware, certificados digitais, e cada vez mais está a aumentar o desenvolvimento para o uso de autenticação biométrica [8].
II. **Distribuição da informação acerca da autenticação e autorização.** Nesta parte do problema o servidor que autenticou o utilizador, ou fornecedor de identidades (IdP), tem de conseguir partilhar essa informação de autenticação sobre o utilizador, ou seja, o utilizador autenticou-se com sucesso (autenticação), juntamente com os detalhes correspondentes à informação que o referido utilizador pode ter acesso (autorização) para qualquer outro serviço interessado.

Por exemplo, pode-se querer informar o serviço X que o utilizador se autenticou com sucesso junto do servidor de autenticação e, portanto, o acesso a X por este utilizador deve ser autorizado. É preciso não esquecer que esta troca de informações entre o servidor de autenticação e o serviço X deve acontecer de forma segura. Caso contrário, não é possível garantir a integridade da autenticação.

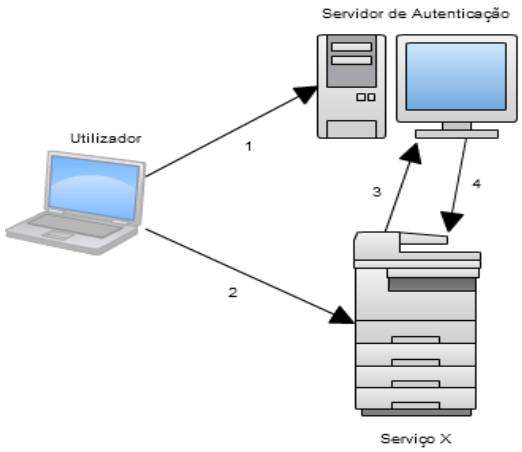

Figura 2 - Autenticação num fornecedor de serviços com recurso a um fornecedor de identidades

1. O Utilizador fornece ao Servidor de Autenticação os dados necessários à sua autenticação (ex.: User-Password).
2. O Utilizador pretende utilizar o Serviço X, como tal estabelece uma ligação com este.
3. O Serviço X verifica junto do Servidor de Autenticação se o utilizador está autenticado e tem autorização para aceder aos recursos pretendidos.
4. O Servidor de Autenticação entrega a resposta ao Serviço X, esta resposta correspondente ao resultado da autenticação feita pelo utilizador em 1.

O exemplo referido pela Figura 2 apresenta um único fornecedor de serviços, pelo que a arquitetura presente na mesma figura foi desenhada com o objetivo de permitir ao utilizador a autenticação num único fornecedor de identidades. Assim, aquando o acesso a diferentes serviços, desde que estes suportem o respetivo fornecedor de identidades, a autenticação pode permanecer válida, fazendo com que o utilizador não precise de colocar novamente os dados da autenticação. Podemos ver também uma melhoria na gestão de contas de acesso do utilizador, sendo que o utilizador precisa apenas de se autenticar perante um fornecedor de identidades, como tal um "token" de acesso ou um conjunto nome de utilizador palavra-chave são necessários para múltiplos serviços.

Nos últimos anos, esta arquitetura levou ao desenvolvimento de um protocolo baseado em XML com o propósito de ajudar a criar soluções de gestão de identidade, no âmbito deste trabalho, SAML [4] (descrita em mais pormenor na secção 2.1.8), de notar também que a arquitetura apresentada na Figura 2 é representativa do princípio de Single Sign On.

2.1.7 Single Sign On (SSO) e Web SSO

O Single Sign On foi um dos principais objetivos para o qual SAML foi criado. Neste ponto pretende-se mostrar o funcionamento genérico da troca de mensagens e a sua interação com o protocolo HTTP, com especial enfase ao "Profile" conhecido como SAML 2.0 Web Browser SSO.

No modelo tórico, SSO tem como principais entidades: i) o navegador que representa o utilizador no decorrer do processo de SSO, ii) o Recurso que contem os dados de acesso restrito requeridos pelo utilizador, iii) o IdP que autentica o utilizador e, iv) o SP que executa o processo de SSO para obter o Recurso. O funcionamento entre estas entidades pode ser descrito em seis passos:

I. O utilizador pretende aceder a um recurso protegido. Este verifica se o utilizador possui uma sessão e se esta é valida. Caso não tenha, o recurso é enviado para o SP para que se dê inicio ao processo de SSO.
II. O SP emite um pedido de autenticação. Por norma o recurso está instalado no mesmo local físico que o SP, para o utilizador poder autenticar-se junto do IdP. Neste momento existe um processo de descoberta do IdP que pode ser feito através de informação no SP apresentada ao utilizador, ou o SP pode delegar a escolha de um IdP a um serviço centralizado existente para o efeito.
III. O utilizador é autenticado no IdP. Quando o pedido chega ao IdP, este verifica o pedido a fim de validar a existência de uma sessão por parte do utilizador no IdP. Caso não se verifique a existência de sessão, o IdP apresenta a possibilidade ao utilizador deste efetuar autenticação através de um dos mecanismos existentes (credenciais de acesso ou certificados).
IV. Após autenticar o utilizador, o IdP emite uma Resposta à autenticação (Authentication Response) e envia esta de volta ao SP.
V. Quando o utilizador volta para o SP com a resposta do IdP, o SP irá validar esta resposta, criar uma sessão para o utilizador e extrair mais alguma informação desta resposta, como por exemplo, o identificador do utilizador, disponibilizando esta mesma informação ao recurso protegido requerido inicialmente pelo utilizador. Depois destes passos serem resolvidos com sucesso, o recurso pretendido fica disponível ao utilizador.
VI. Neste momento o utilizador volta a tentar aceder ao recurso, possuindo uma sessão válida criada na secção anterior. Assim o recurso sabe quem o utilizador é e se tem privilégios para aceder à informação. Com esta informação o recurso decide se pode ou

não entregar a informação pedida pelo utilizador. Em caso afirmativo, é entregue ao utilizador a informação pretendida por este inicialmente.

Todas as interações entre as entidades da conversação, à exceção de III, podem usar um dos seguintes métodos para trocar mensagens entre si:

- HTTP POST binding: define como as mensagens do protocolo SAML podem ser transportadas no conteúdo codificado em base64 de um FORM de controlo HTML.
- HTTP Artifact binding: define como um ARTIFACT é transportado de quem envia a mensagem para quem a recebe utilizando o protocolo HTTP para o transporte. Dois mecanismos são fornecidos para que tal aconteça: um formulário de controlo HTML ou na mensagem de pedido presente no URL.
- HTTP Redirect binding: define como as mensagens do protocolo SAML podem ser transportadas utilizando mensagens de redireccionamento por HTTP (resposta com o código 302).

Para a interação descrita em III, o método de HTTP Redirect binding não pode ser usado uma vez que, devido ao tamanho da resposta, esta possa ultrapassar o tamanho máximo permitido para uma mensagem HTTP Redirect [18].

2.1.8 SAML

Security Assertion Markup Language (SAML) [4], é um padrão aberto que define o formato de dados para troca de informações relativas a autenticação e autorização entre serviços, mais particularmente entre um fornecedor de identidades e um fornecedor de serviços. É um produto da OASIS Security Services Technical Committee que data de 2001 com a mais recente atualização em 2005 para a versão 2.0 utilizada atualmente.

SAML veio responder ao problema de "single Sign On" (SSO) presente na secção2.1.6, ao nível da Internet para acessos dentro do mesmo domínio existem muitas soluções deste género (por exemplo, utilizando cookies), no entanto expandir essas soluções para irem além do mesmo domínio tem-se mostrado ser um problema de difícil resolução, que acabou por levar ao desenvolvimento de tecnologias como SAML.

O desenho da "Framework" SAML permite que os seus componentes sejam combinados, suportando uma grande variedade de cenários. Esta framework consiste em três componentes chave: *core*, *bindings*, p*rofiles*.

i. Core: consiste nas *asserções de segurança*, que definem a sintaxe e a semântica das mensagens, e os protocolos gerais para transferência das mesmas. Neste contexto,

asserções são pacotes de dados em XML que transportam expressões SAML com os dados sobre o utilizador. Este conjunto de dados pode conter muito mais que os dados de autenticação do utilizador. Pode conter por exemplo como foi feita a autenticação ou mesmo a data em que o utilizador se autenticou. O protocolo de pedido/resposta, é também especificado em XML. Neste protocolo um "Request" é um pedido do tipo "query" de uma asserção e a resposta deve devolver a asserção ou um erro. Ao processo de encapsular o Core, mensagens e protocolos num protocolo subadjacente, é chamado Binding.

ii. Bindings: especifica como podem as mensagens ser mapeadas em outros protocolos comuns como Simple Object Access Protocol (SOAP) [18] ou HTTP. Utiliza protocolos de transferência de mensagens para permitir aos sistemas que suportam SAML transferir mensagens de forma segura. No entanto, SAML, ou um dos protocolos na comunicação, devem suportar autenticação mútua, integridade de mensagens e confidencialidade com o protocolo SSL.

iii. Profiles: especifica como o Core e Bindings são utilizados numa aplicação específica. Por exemplo, o perfil de Web Browser SSO, utiliza o protocolo de pedido de autenticação com Binding por HTTP e SOAP, e, apesar de existirem muitos perfis de SAML, em SSO a especificação de perfis inclui apenas Web Browser SSO Profile, Enhanced Client ou Proxy (ECP) Profile, Identity Provider Discovery Profile, Single Logout Profile e Name Identifier Management Profile. De notar que estre perfis não são equivalentes. Perfis diferentes desempenham funções diferentes e que, apesar de SAML definir muitos perfis, Web SSO foi a principal razão pela qual o padrão foi criado.

A Figura 3 mostra a relação existente entre os componentes SAML descritos anteriormente, em que o componente Profiles descreve o contexto em que o SAML é usado e Bindings especifica o protocolo usado para encapsular as mensagens SAML. Bindings e Profiles são baseados no Core que define o formato das mensagens SAML e os protocolos genéricos de pedido/resposta.

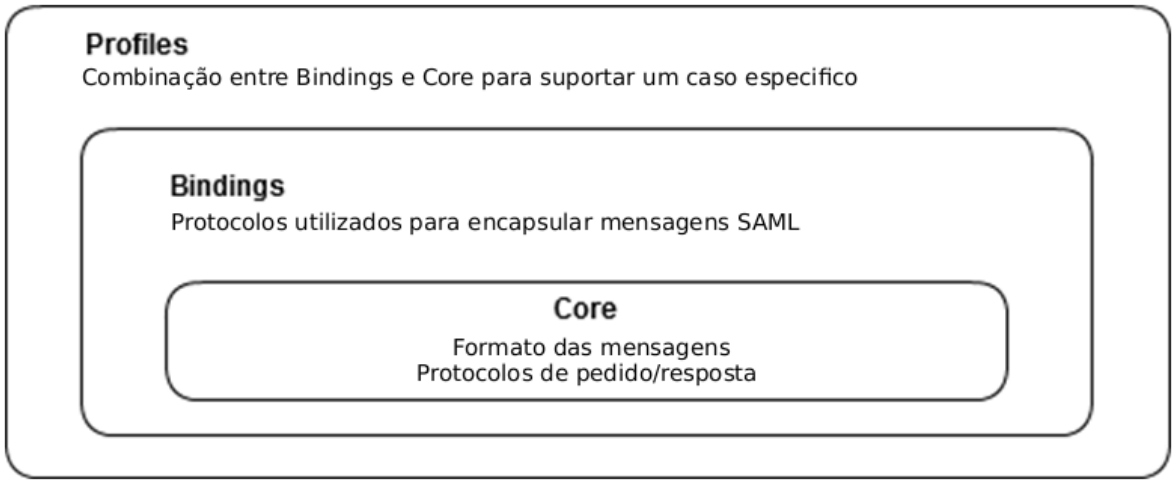

Figura 3 - Relação entre Profiles, Binding e Core na framwork SAML

Como mecanismo de autenticação, a especificação SAML define três entidades principais:

i. O "principal", normalmente o utilizador.
ii. O fornecedor de identidades (IdP)
iii. O fornecedor de Serviços (SP)

Estas três entidades atuam entre si da seguinte forma, o utilizador efetua um pedido de um determinado serviço ao SP, este por sua vez pede ao IdP uma asserção de identidade e recebe a resposta. Com base nessa asserção o SP toma uma decisão de controlo de acesso, ou autorização, e permite ou não o acesso ao recurso pedido pelo utilizador anteriormente.

Antes de entregar a asserção de identidade ao SP, o IdP pode pedir informação adicional ao utilizador, esta informação é referente à autenticação do utilizador podendo por exemplo ser usadas credenciais de autenticação ou assinatura com par de chaves, para autenticar o mesmo. Em SAML um IdP pode fornecer asserções SAML para muitos SP e, ao mesmo tempo, um SP pode confiar em vários IdP para fornecer identidades acerca de utilizadores.

O SAML não especifica como deve ser implementado o fornecedor de identidades, pelo que, o fornecedor de identidades pode usar um sistema de utilizador/palavra-chave, credenciais ou outras implementações desenvolvidas para o efeito.

Muitos produtos de gestão de identidades existentes hoje em dia suportam SAML, tanto que sistemas como Internet2 [18], Shibboleth (http://shibboleth.net/), Liberty Alliance (http://www.projectliberty.org/) e OASIS Web Services Security (WS-Security). Finalmente, uma outra aproximação mais recente ao problema de Browser SSO é o protocolo OpenID [5].

2.1.9 OPENID

O OpenID é um protocolo descentralizado para gestão de identidades e "Single Sign On" (SSO) na Internet. Surgiu como uma resposta eficaz ao problema da gestão de identidades digitais referido em 0. Este protocolo permite aos utilizadores autenticarem-se em diferentes sítios na Internet usando para o efeito uma única identidade digital, que eles próprios podem gerir. Desta forma elimina-se a necessidade de criar diferentes nomes de utilizadores e passwords para cada site, podendo passar a usar-se apenas um OpenID para todos os processos que necessitem de serviços de identidade devidamente autenticada na Internet. O "login" por meio do OpenID, permite aos utilizadores registarem-se junto de um fornecedor de identidades apropriado, onde poderão ser eles próprios a gerir as características da identidade digital com que se quer dar a conhecer a um serviço. No momento da autenticação para utilizar dados específicos num determinado serviço, o utilizador pode escolher quais dos dados que guardou anteriormente divulgar ao serviço.

Como o serviço é completamente descentralizado, uma vez que qualquer site pode usar o OpenID como forma de "sign-in" e os utilizadores são livres de escolher o provedor de identidades que melhor se adeque ao seu perfil de atividade ou serviço na Internet, podendo mesmo mudar de fornecedor de serviços. Desta forma é possível passar a ter mais confiança nos serviços e na gestão dos dados que constituem a identidade digital de um utilizador.

Para utilizar um serviço que suporte OpenID o utilizador tem apenas de registar a sua informação pessoal com o fornecedor de identidades e escolher um URL (Uniform Resource Locator) ou XRI (Extensible Resource Identifier). Como exemplo poderia ser tomas.openid.experiencia.pt.

O funcionamento deste protocolo é algo simples. Por exemplo, um utilizador oferece ao fornecedor de serviços o seu identificador OpenID, no exemplo anterior tomas.openid.experiencia.net, ao que o fornecedor de serviços converte para a forma canónica, ou endereço URL, ficando assim HTTP://tomas.openid.experiencia.pt, este endereço é utilizado pelo fornecedor do serviço para estabelecer a comunicação com o fornecedor de OpenID, uma vez que o mesmo é reconhecido como sendo um endereço válido desde que esteja devidamente registado.

Há duas formas que podem ser usadas para estabelecer a identidade do utilizador e disponibilizar os dados de um utilizador com OpenID:

I. checkid_immediate, no qual o fornecedor de serviços força a que o fornecedor de OpenID não interaja com o utilizador, fazendo com que todo o processo de comunicação seja

encaminhado pelo “user-agent” ou UA do utilizador, sem que este tenha conhecimento explicito que existe uma comunicação entre o fornecedor de serviços e o fornecedor de identidades.

II. checkid_setup, em que o utilizador comunica diretamente com o fornecedor de OpenID através do mesmo UA utilizado para aceder ao fornecedor do serviço.

Caso a operação em modo de “checkid_immediate” não possa ser automatizada por um qualquer motivo, este pode reverter para “checkid_setup”.

No funcionamento deste esquema, em primeiro lugar o fornecedor do serviço e o fornecedor de OpenID podem opcionalmente estabelecer um segredo partilhado, designado por “associate handle”, que o fornecedor do serviço guarda para gerar mais segurança na transferência de informação entre estes. Se o modo utilizado for “checkid_setup”, o fornecedor do serviço redireciona o UA do utilizador para o fornecedor de OpenID, para que este se possa autenticar diretamente com o OpenID utilizando as suas credenciais de acesso. Estas credenciais de autenticação podem variar, mas normalmente são utilizadas credenciais como palavra-chave ou certificado. Após receber estas, o fornecedor de OpenID questiona o utilizador para certificar se este confia no fornecedor de serviços que está a efetuar o pedido da informação específica.

Se o utilizador rejeitar este pedido de informação, é devolvida uma mensagem contendo a informação que o pedido foi rejeitado. Caso o utilizador aceite, é devolvido ao fornecedor do serviço a asserção de autenticação do utilizador para este confirme se a autenticação foi efetuada no devido fornecedor de OpenID e teve sucesso [14].

2.1.10 Shibollet

O trabalho Shibboleth começou como “Internet2 Middleware” em 2000. Este trabalho visava trazer um novo tipo de serviços e funcionalidades à web na data de iniciação. Mais tarde, nesse mesmo ano, o trabalho ligou-se ao grupo de trabalho OASIS SAML, utilizando então desde o seu início, SAML. Shibboleth 1.0 foi lançado em 2003. Com o lançamento de SAML 2.0 em 2005 e Shibboleth 2.0 no ano seguinte, o padrão representado por SAML cresceu e passou a incluir todas as abordagens referentes a meta-dados desenvolvidas inicialmente por Shibboleth.

O Shibboleth foi desenhado para ser utilizado como um sistema padrão baseado em “software” de código aberto para web single sign-on com o propósito de ser utilizado dentro de uma empresa, ou até mesmo entre empresas.

Este sistema permite que os fornecedores de serviços tomem decisões acerca do acesso à informação por parte de um utilizador ou serviço, mantendo sempre o princípio da preservação da privacidade do utilizador.

Como software, o Shibboleth implementa os padrões utilizados em autenticação federada, notoriamente presente no conceito de segurança pela utilização de OASIS 'Assertion Markup Language, que permite fornecer serviços de entrega de atributos e autenticação através de single sign-on.

Shibboleth também fornece funcionalidades de privacidade, permitindo que os diferentes navegadores e respetivos utilizadores controlem os dados disponibilizados a cada aplicação ou serviço. Este software é amplamente utilizado atualmente para várias funções, mas de uma forma geral, tem como principal objetivo garantir o acesso e gerir a identidade dos utilizadores e aplicações ou serviços. Shibboleth é desenvolvido em open-source e está disponível gratuitamente sob a licença Apache Software. O seu funcionamento é muito semelhante ao de um web SSO fornecendo a componente de um SSO federado.

O funcionamento de SSO, como descrito na secção2.1.7, define que o fornecedor de identidades e o fornecedor de serviços têm de estar dentro do mesmo domínio. SSO é designado de Federado ou web SSO quando o fornecedor de identidades e o fornecedor de serviços estão em domínios diferentes, pois faz sentido que um serviço possa usar diferentes IdP's em domínios diferentes do seu.

2.2 Problemas de Sessões HTTP e SSL

2.2.1 Problemas de sessões HTTP

Neste contexto, uma sessão aplicacional surge quando um cliente interage pela primeira vez com um servidor, fazendo com que este crie uma sessão HTTP para saber que quando trocar qualquer tipo de dados (pedido de credenciais, informação de inicio de sessão, etc) estará a fazê-lo com o cliente correto e não um qualquer outro que se ligou posteriormente a esse mesmo servidor.

Normalmente, esta informação é guardada em "Cookies" que são geridos pelo navegador, no caso de ser um utilizador, ou por outros dados como origem do pedido, endereço de IP, ou outros. O maior problema num caso destes é que a informação de sessão pode ser facilmente roubada ou manipulada, por forma a fazer com que o servidor pense que está a comunicar com o cliente pretendido, mas na realidade está a comunicar com um segundo cliente preparado com toda a informação referente ao primeiro cliente da comunicação, cuja informação fora

inadvertidamente roubada por “session hijack”. Desta forma, na troca de informação entre utilizadores e serviços “web”, um dos grandes problemas encontrado nas sessões HTTP prende-se com a segurança na autenticação dos utilizadores e respetivo acesso a serviços, em que mesmo que exista um controlo de acesso de utilizadores, é muito difícil certificar a entidade com quem se deseja comunicar. Desta forma, é difícil ao cliente ter a certeza que o serviço que pretende está a ser fornecido pela entidade correta, assim como é difícil ao fornecedor do serviços ter a certeza que está a comunicar realmente com o cliente correto e não com um cliente impersonificado.

Uma vez que este tipo de ataques é bem conhecido e, como resultado dos mesmos, se consegue de uma maneira simples obter os dados de autenticação do um cliente, este é sem dúvida um dos problemas que o servidor de autenticação terá numa autenticação em sessões HTTP.

Da mesma maneira que o servidor pode ser enganado a pensar que está a comunicar com o cliente/utilizador correto, também o utilizador pode ser levado a pensar que está perante o serviço correto e fornecer os seus dados pessoais a entidades terceiras não confiáveis para tal informação [15].

2.2.2 Problemas de sessões SSL

Uma sessão SSL/TLS no contexto web, corresponde a uma ligação HTTPS com um determinado serviço. Este tipo de sessões implica normalmente que o serviço possua um certificado de chave pública (2.1.3) que o navegador do utilizador, ou cliente, considere como válido. Para a maioria dos serviços existentes na atualidade, esta autenticação é apenas considerada num sentido, do cliente para o servidor, ou seja, para o serviço é irrelevante quem a este se está a ligar, sendo o que importa é que o utilizador possa assegurar e certificar o serviço que pretende utilizar. Esta medida previne que sites extremamente semelhantes aos originais induzam os utilizadores de serviços a fornecer dados pessoais críticos a uma outra entidade não confiável. Este processo tem a designação de “site phishing”.

Na forma como o sistema está implementado o utilizador confia no serviço para lhe fornecer os seus dados pessoais, (como nome de utilizador e palavra-chave), para que o serviço o autentique posteriormente e forneça a informação desejada, ou certifique a sua autenticação com um outro serviço. No entanto, tendo em conta o problema de “site phishing” existe a possibilidade do utilizador se encontrar no site correto e válido para o serviço que pretende, mas devido a um vírus ou software malicioso presente no seu computador lhe serem pedidos dados adicionais (que normalmente não seriam necessários) com os quais os atacantes podem extrapolar mais informações ou mesmo prejudicar o utilizador. Considerando o caso de acesso a

um serviço de “net banking” referido como exemplo na secção 2.1.3, um utilizador liga-se ao site da Caixa Geral de Depósitos para efetuar uma consulta à sua conta, ao que o seu navegador certifica a página e o serviço como sendo válidos. Mas o computador deste utilizador está comprometido com um software que, depois de ter colocado os seus dados de login, lhe pede que coloque também a totalidade dos valores presentes no seu cartão matriz para validação do seu acesso. Este utilizador, não sabendo o porquê deste pedido, pensa que será um procedimento de rotina, quando na verdade estes dados iram diretamente para o atacante que, posteriormente, terá acesso à sua conta e a possibilidade de efetuar qualquer movimento ou transação. Esta última consequência seria evitada se o utilizador possuísse uma chave pública num “smartcard” e o serviço requeria a sua utilização quando necessário, ao que o utilizador teria sido previamente informado que era suficiente para efetuar qualquer transação, não precisando de utilizar qualquer nome de utilizador ou palavra-chave.

Embora um pouco fora de contexto com este trabalho, pretende-se demonstrar a importância do uso de certificados de chave pública do lado do utilizador/cliente, e consequentemente, a autenticação mútua para uso de serviços críticos, como é o acesso a uma conta bancária e transações online dentro do banco.

Este tipo de sessões tem um problema do ponto de vista do programador, uma vez que têm pouco controlo ao nível da sessão SSL/TLS, ou embora seja possível estabelecer uma sessão TLS e existir a possibilidade de extrair o certificado apresentado por cada uma das entidades na conversação, não é possível remover ou terminar a sessão segura mantendo a sessão web ativa. Um exemplo seria a possibilidade de nos ligarmos a um serviço que, não precisando de autenticação, nos fornecia alguns dados, e para outros tipos de dados era exigida autenticação. Assim, o utilizador precisaria de se autenticar num serviço apenas quando os dados que requeria fossem críticos ou seguros, podendo navegar entre estes dados sem necessidade de reestabelecer uma sessão relacional, estabelecendo apenas a sessão segura usando necessário. Este conceito vem de encontro ao problema presente na secção 2.2.

2.2.3 Sessão SSL/TLS versus Sessão HTTP

A secção 2.2.1 e a secção 2.2.2 mostram alguns dos problemas conhecidos na utilização de serviços online. Existe uma dependência clara entre eles. Em concreto, para que um canal SSL/TLS possa ser criado, é necessário que o cliente e servidor troquem informação entre si. Esta informação não é cifrada e, durante este processo, a comunicação é feita sobre HTTP, pelo que deverá existir uma sessão não segura. Quer-se com isto dizer que existe a necessidade de

existência de uma camada HTTP com a qual é então criada a camada SSL/TLS, a fim de reforçar a segurança durante a comunicação. Para que um sistema possa comunicar seguramente com um outro, estes dois sistemas têm de iniciar um diálogo entre eles. Num dos sistemas, o cliente, pretende-se aceder a um conjunto específico de informação e, no outro, o servidor detém a informação pretendida pelo cliente. Existe como mecanismo de segurança o envio de uma resposta informando o cliente que terá de estabelecer um novo canal num determinado protocolo, enviando-o para o endereço correto. Este processo é designado de reencaminhamento e acontece, normalmente, sempre que um utilizador ou serviço tenta aceder por HTTP (aplicacional) a um recurso na internet que requer uma sessão HTTPS (TLS).

As sessões TLS, apesar de fornecerem um nível de segurança mais elevado, tanto por cifrarem a comunicação entre o cliente e o servidor, têm algumas limitações no que diz respeito à gestão das mesmas assim que um cliente estabelece uma sessão com o servidor, sendo que, sempre que um utilizador estabelece uma sessão deste tipo, as operações de controlo que um programador tem sobre o controlo da sessão SSL são limitadas.

No âmbito deste trabalho procedeu-se à verificação de vários servidores web e correspondentes API de controlo de sessões SSL, API's estas demonstradas na secção 3.1, pelo que se percebeu que a maioria dos servidores mais utilizados como Tomcat [28], Apache [30] e IIS [29] permitem um controlo básico da sessão. É possível então obter o ID da sessão, o certificado do cliente caso necessário para autenticação mútua e definir que certificados aceitar, definir os parâmetros como o protocolo, a gestão das chaves de sessão, entre todos os demais parâmetros necessários à sessão. No entanto, não é possível em nenhum deles fechar uma sessão SSL sem que para isso seja necessário desligar toda a comunicação com o cliente. Ou seja, é necessário fechar toda a comunicação e estabelecer uma nova se o utilizador apenas pretendesse trocar a autenticação utilizada ou mudar o tipo de autenticação para por exemplo aceder a um recurso diferente, mais seguro [16]. No servidor Tomcat v7 é possível exercer o comportamento anterior, fechar a sessão SSL sem desligar completamente o cliente, no entanto o comportamento dos navegadores testados mostrou-se errático não se comportando sempre da mesma forma [19].

2.2.4 Teste a diferentes navegadores Web

Antes do início da realização deste trabalho optou-se por instalar diferentes navegadores de acesso à Internet, nomeadamente, FireFox [24], Internet Explorer (IE) [25], Google Chrome (GC) [26] e Opera [27]. A escolha de vários navegadores fica a dever-se em primeiro lugar à importância de que diferentes pessoas utilizam diferentes navegadores, pelo que é importante testar o comportamento dos vários a fim de obter um melhor resultado possível em diferentes

condições. Em segundo lugar, como são navegadores diferentes, apesar de haver normas pelas quais estes se devem guiar para serem considerados navegadores da “web”, o resultado de uma aplicação ou serviço “web”, no que diz respeito ao processo de autenticação, pode não ser o mesmo se, por exemplo, esta aplicação for executada no Firefox ou no Opera.

Navegador	Autenticação com SmartCard	Remoção do Smartcard
Opera	Não	N/A
FireFox	Sim	Deteta
Google Chrome	Sim	Deteta
Internet Explorer	Sim	Não deteta

Tabela 1 - Navegadores e suporte para autenticação com SmartCards

Como se pode ver pela Tabela 1 dos navegadores testados, apenas o Opera não apresentou suporte para autenticação “online” com "SmartCard", os três restantes, FireFox, IE e GC apresentam suporte para este tipo de autenticação.

Os testes iniciais nos diferentes navegadores foram todos em sites portugueses uma vez que o CC, como sendo para o uso nacional, apenas é reconhecido como meio de autenticação online em Portugal ou nos sites portugueses que o permitam.

Após o teste em alguns sites de autenticação, nomeadamente o serviço de finanças português (http://www.portaldasfinancas.gov.pt/pt/home.action), foi percetível que a resposta dos diferentes navegadores era diferente em diferentes circunstâncias, à data da experiencia. A título de exemplo, num dos navegadores era possível retirar o CC após a autenticação e continuar a navegar, num outro existia um pedido constante de PIN (Personal identification number) mesmo após o PIN correto ter sido inserido.

Durante estes testes notou-se uma anomalia na utilização do referido site das finanças português. Após se efetuar o login com um CC era possível aceder aos conteúdos e serviços do titular do cartão como seria de esperar. No entanto, depois de se efetuar o “logout” através do botão para o efeito na página, ao voltar a iniciar de imediato uma nova ligação, efetuando para isso a respetiva autenticação, com um CC pertencente a um segundo indivíduo, o titular que aparecia autenticado era o do cartão antigo. Ou seja, apesar da ligação ser segura e exigir autenticação mútua, a sessão do utilizador é mantida através de uma sessão HTTP, fazendo com que, apesar de um novo utilizador estar a autenticar-se no mesmo navegador que um outro anteriormente (o processo de autenticação é sempre obrigatório na autenticação mútua como referido nas secções 2.2.2 e 2.1.5), o nome e a conta a que este utilizador acede são as do

utilizador anterior. Isto acontece porque a sessão HTTP anterior ainda não expirou, o servidor está preparado para aceitar sessões criadas anteriormente fazendo com que este problema aconteça. Esta informação foi enviada de imediato pelo Professor Dr. André Zúquete para os serviços competentes.

3 Controlo de sessões SSL por servidor HTTP

De acordo com o descrito na secção 1.2, este trabalho foi pensado inicialmente com o objetivo de explorar a capacidade dos diversos servidores web, durante os processos de autenticação, de-autenticação e reautenticação de um utilizador através do uso das credencias de autenticação presentes no CC (secção 2.1.4).

É pretendido então compreender o comportamento dos diferentes servidores web a fim de se descobrir a existência de algum mecanismo nos servidores, respetivas linguagens de programação e API's, que permita controlar na sua totalidade uma sessão SSL/TLS, protocolo utilizado para estabelecer uma sessão HTTPS.

3.1 Servidores Web

Deu-se então início ao processo de pesquisa das API's apresentadas pelos vários servidores web testados. Foram verificadas as API's dos seguintes servidores, Tomcat V6, Tomcat V7, IIS e Apache. Todos estes servidores são capazes de fazer a gestão de certificados do seu lado. Isto é, após a criação de um par de chaves e o respetivo certificado de chave pública, qualquer um dos servidores está preparado para criar uma sessão SSL com o cliente, permitindo autenticação mútua, pelo que aceitam e verificam o certificado de chave pública enviado pelo cliente, a verificação é sempre feita de acordo com o processo descrito na secção 2.1.5.

O maior problema encontrado foi no acesso à sessão SSL criada e o controlo da mesma do lado do servidor. É pretendido que, depois de ter acesso à sessão, poder invalidar a mesma de forma a forçar uma reautenticação do utilizador e, se possível, evitando o processo inicial de troca de parâmetros para estabelecer a sessão, tornando assim o processo de reautenticação mais rápido que o inicial de autenticação. Os resultados obtidos estão demonstrados na Tabela 2.

Servidor Web	Permite no controlo da Sessão SSL	Não permite no controlo da Sessão SSL
Apache 2.0	Não permite qualquer controlo da sessão. É possível aceder a alguns dados como "sessionID" ou certificado enviado pelo cliente através do uso de Java Servlets.	Não é possível invalidar a sessão ou forçar a reautenticação do cliente. Sem acesso à camada protocolar SSL/TLS.
IIS	É possível aceder ao certificado do cliente e forçar alguns parâmetros para o início da sessão através da linguagem C#. É possível terminar toda a comunicação.	Não é possível forçar uma renegociação ou invalidar a sessão. Sem acesso à camada protocolar SSL/TLS.
Tomcat V6	Através do uso de Java Servelts é possível aceder ao certificado de chave pública, estabelecer alguns parâmetros iniciais da sessão e obter o "sessionID".	Não é possível forçar uma renegociação, ou invalidar a sessão. Sem acesso à camada protocolar SSL/TLS.
Tomcat V7	Comportamento igual ao Tomcat V6. Permite algum acesso à camada protocolar SSL. Invalidar a sessão e enviar um cabeçalho ao cliente a informar o mesmo.	Acesso limitado à camada protocolar. Não permite forçar apenas a reautenticação.

Tabela 2 - Controlo de acesso a uma sessão SSL em vários servidores Web

Como é possível verificar pela Tabela 2, apesar da maioria dos servidores web existentes atualmente fornecerem mecanismos de autenticação mútua e gestão de parâmetros de sessões SSL, o controlo deste tipo de sessões é muito limitado ou mesmo inexistente, pelo que é extremamente difícil, ou mesmo impossível com os mecanismos existentes, ter um cliente autenticado num serviço e pedir a sua reautenticação, por exemplo, numa mudança de serviço. De notar que, para reautenticação válida, é necessário voltar a pedir os dados de autenticação do utilizador.

Uma vez que o servidor Web Tomcat V7 permite algum controlo da sessão SSL, foram efetuados alguns testes neste, com o objetivo de perceber as potencialidades de controlo de uma sessão SSL com o mesmo. Os resultados obtidos serão demonstrados na secção seguinte (3.2).

3.2 Navegadores

Tendo em conta que numa sessão segura os navegadores Web (Browser's) são um componente também importante, uma vez que estes possuem mecanismos de validação de certificados para autenticação mútua com os servidores. Foram efetuados alguns testes a estes e às potencialidades da API do Tomcat V7, utilizando para isso um servidor Web básico criado na linguagem Java recorrendo a Java EE (Servlets). Este servidor recebe um pedido HTTP Get, normalmente utilizado para obter um recurso de um servidor "web". Se o cliente não estiver autenticado reencaminha este para uma página de autenticação iniciando o processo de

autenticação mútua, depois de efetuada a autenticação o servidor inicia o processo de invalidação de sessão. As características do "hardware" utilizado para os testes estão descritas na secção 6.2.

Seria de esperar que, apesar da utilização de navegadores diferentes, o resultado fosse semelhante em todos eles. No entanto, isto não se verifica, pelo que os resultados obtidos com esta API são bastante diferentes nos vários navegadores. De notar que o navegador Opera não foi testado, pois como indicado na secção 2.2.4 não tem suporte para utilização de smartcards.

Navegador	**Resposta ao processo de invalidação de sessão.**
FireFox	Termina corretamente a sessão SSL, mantendo uma sessão HTTP ativa e pedindo ao cliente reautenticação quando este tenta novamente aceder a um recurso protegido. No entanto não reconhece a sessão anterior e aceita a autenticação do cliente criando uma nova sessão SSL.
Google Chrome	Comportamento igual ao Firefox
Internet Explorer	Termina corretamente a sessão SSL, mantendo uma sessão HTTP ativa e pedindo por vezes ao cliente a reautenticação quando este tenta aceder a um recurso protegido novamente, quando não pede novamente a autenticação ao cliente permanece num estado em que não aceita qualquer pedido de autenticação nem permite acesso a qualquer recurso.

Tabela 3 - Resposta dos diferentes navegadores ao processo de invalidação de uma sessão SSL com Tomcat V7

Como se pode ver na Tabela 3, a resposta à API utilizada no Tomcat V7 não é em nada viável para uma implementação sendo que não se pode esperar uma resposta comum entre diferentes navegadores. Podemos concluir com os testes e experiencias efetuadas que existe a necessidade de criar um mecanismo diferente para retirar a dependência do processo de autenticação de uma sessão SSL dos servidores e navegadores "web".

Com os resultados obtidos na análise dos diferentes servidores "web", é possível verificar a importância de um mecanismo, que permita a não dependência dos mesmos para a utilização de diferentes métodos de autenticação.

Pela análise dos diferentes navegadores "web", é fácil verificar a importância de um mecanismo independente dos mesmos que permita a autenticação de utilizadores, recorrendo à utilização de "smartcards", neste caso o CC.

Assim, tendo como base a análise efetuada nesta secção, compreende-se o porquê da exploração de uma arquitetura de autenticação diferente das demais, em que o acesso à informação presente no CC é da competência de uma aplicação. A mesma deverá ser executada no equipamento do utilizador. Esta arquitetura encontra-se detalhada na secção seguinte (4).

4 Autenticação ao nível aplicacional (HTTP) com CC

A pesquisa e resultados obtidos na secção 3 levou a que se desenvolvesse um IdP para utilizar no equipamento do cliente a fim de tornar possível fazer uma gestão eficaz da autenticação, e respetivos processos de reautenticação entre o cliente e o servidor retirando assim a dependência dos servidores Web e/ou navegadores Web. Este IdP foi designado de CHelper.

O CHelper é um serviço que corre no equipamento do cliente, contendo um pequeno servidor Web à escuta de ligações vindas da máquina local numa determinada porta. É pretendido para este novo tipo de IdP que qualquer SP, desde que aceite a cadeia de certificação do CC, consiga autenticar um utilizador que pretenda autenticar-se utilizando esse mesmo cartão. Para tal foi desenhada uma arquitetura cujo princípio de comunicação entre componentes e utilizador está representado num diagrama de sequência, presente na Figura 2. Esta arquitetura permite então que o CHelper execute todas as operações necessárias ao CC, para que o detentor do mesmo se possa autenticar perante um fornecedor de serviços e obter os recursos protegidos desejados. É preciso ter em conta que o SP deve estar preparado para este tipo de autenticação, pois deve enviar um conjunto específico de parâmetros ao CHelper. Os parâmetros enviados pelo SP são descritos com mais pormenor na secção 6.1.

No envio do pedido de autenticação ao CHelper, uma vez que é o pedido é efetuado pelo SP existem alguns aspetos no início desta comunicação a ter em conta, sendo que o principal é como encontrar a localização do CHelper, presente no equipamento do cliente, e por conseguinte iniciar a comunicação com este. O CHelper encontra-se detalhado na secção 6.1.

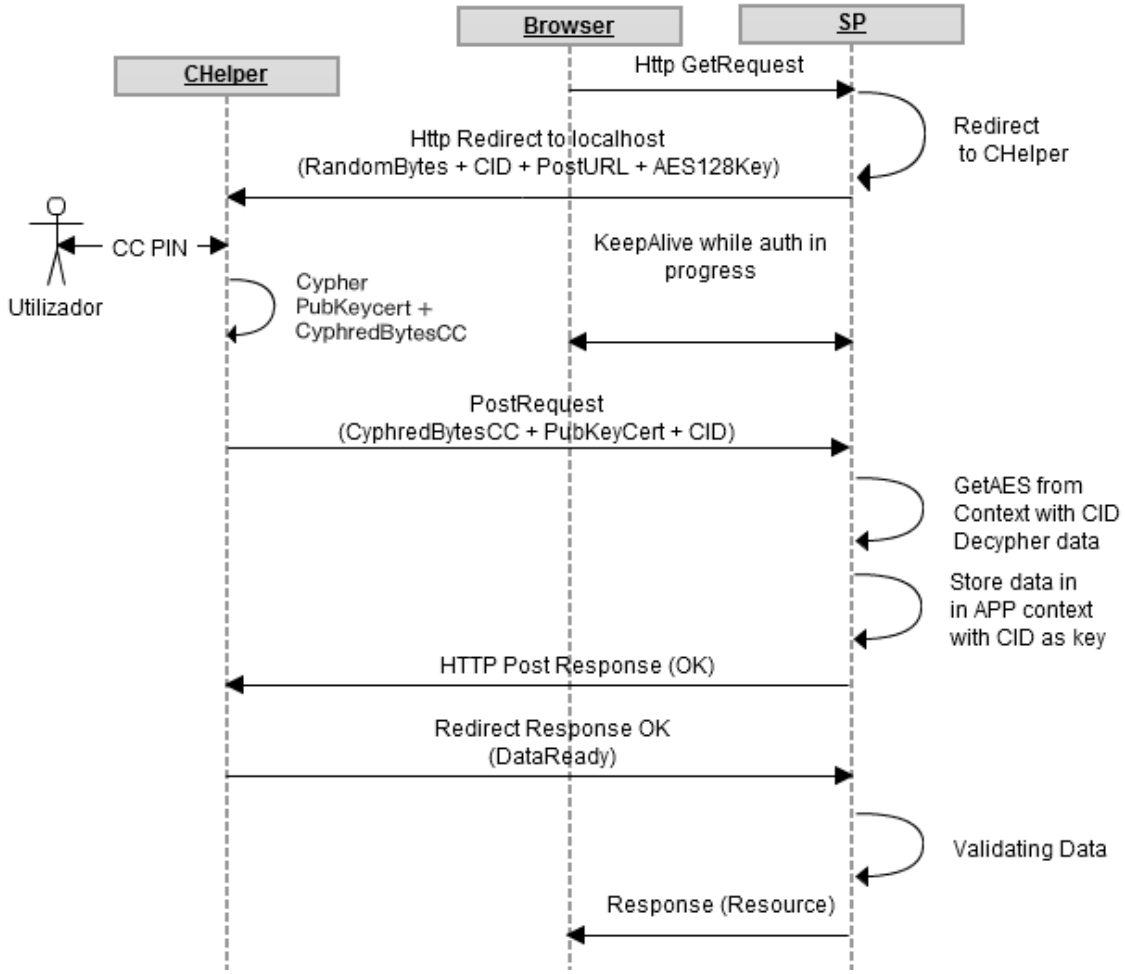

Figura 4- Arquitetura de funcionamento do CHelper

Um qualquer servidor Web pode receber um pedido de um cliente e responder ao mesmo uma vez que o cliente iniciou a comunicação. Por esta razão, é irrelevante se o cliente se encontra por detrás de NAT (Network Address translation) ou uma “firewall”, pois a resposta chegará ao cliente. No entanto isto não acontece se o servidor pretender iniciar ou aceder a um serviço que está a correr no cliente, como é o caso em que o servidor, a iniciar um serviço no cliente, veja o seu pedido ser barrado por uma “firewall” (Figura 5), ou impedido de chegar ao cliente devido a NAT. Para contornar este problema, pensou-se que o ideal seria fazer um HTTP Redirect ao cliente (Browser) em que o destino deste era o "localhost" (Figura 6). Ou seja a própria máquina do cliente faz com que este inicie o serviço pretendido sem a preocupação do pedido iniciado pelo servidor nunca chegar ao cliente, uma vez que é o próprio Browser que se encontra na máquina do cliente a iniciar o serviço. Não sabendo inicialmente o resultado deste processo, o mesmo foi testado nos navegadores utilizados neste trabalho, IE, GC, Firefox e Opera, verificando-se o iniciar do servido no cliente com todos eles.

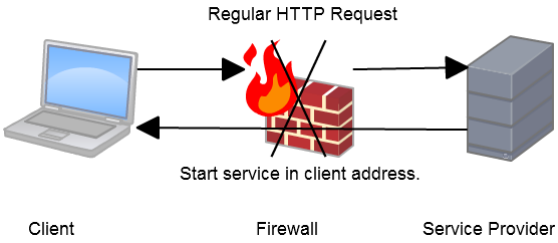

Figura 5 - SP tenta iniciar um serviço no equipamento do cliente através de um novo pedido

Qualquer pedido vindo do SP que não derive de um pedido interno criando anteriormente pelo cliente, será bloqueado, neste caso pela firewall. No caso de NAT será descartado pois o "router" não saberá para onde dirigir o pedido. Neste cenário de rede, atendendo a que o SP tenta iniciar o CHelper através de um novo pedido, seria impossível iniciar o serviço CHelper uma vez que este se encontra a correr no equipamento do cliente.

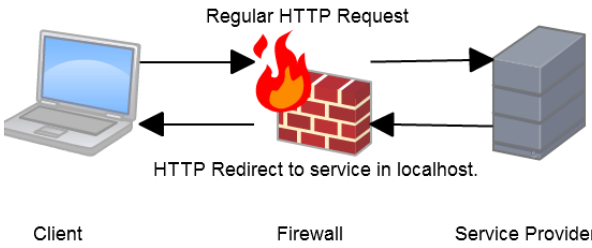

Figura 6 - SP envia um HTTP Redirect como resposta ao pedido do cliente

Sempre que existe um pedido da rede interna, nesta caso, do cliente, este pedido é mapeado e a "firewall" permite que a resposta vinda do exterior seja entregue ao cliente, no caso de NAT o pedido do cliente é mapeado para que o "router" consiga saber a que pedido corresponde a respetiva resposta. Neste cenário de rede, e atendendo a que o SP efetua uma resposta ao cliente com um HTTP Redirect, é perfeitamente viável a chamada a um serviço dentro do equipamento do cliente, neste caso o CHelper.

Para melhor interpretação do diagrama de sequência (Figura 4), a troca de informação entre as entidades intervenientes pode ser descrita da seguinte forma:

A primeira mensagem envida pelo navegador Web (Browser) é um pedido HTTP, em que se pretende obter um recurso protegido do SP. O SP verifica se o Browser está autenticado e, caso não esteja, efetua um pedido de autenticação ao CHelper enviando um HTTP "redirect" com o destino do mesmo bem como os parâmetros necessários ao processo de autenticação (e.g. RandomBytes, Cid, AES128Key), Figura 4.

Ao receber o pedido de autenticação, o CHelper deverá analisar este verificando se é um HTTP GET e contém os parâmetros necessários para efetuar uma assinatura com o CC e à criação

da resposta para o SP. Em caso afirmativo, inicia o processo de comunicação com o CC presente no mesmo equipamento, de onde deve extrair o certificado de chave pública. Deve de seguida assinar o parâmetro RandomBytes enviado pelo fornecedor de serviços e, finalmente, cifrar estes dois parâmetros com a chave AES enviada pelo servidor e enviar um HTTP POST para o SP.

Durante todo o processo precedente ao HTTP POST do CHelper, deve ser sempre mantida a comunicação entre o SP para que não ocorra "timeout" no Browser por falta de resposta do pedido de Redirect, uma vez que todo o processo de extração de certificado e assinaturas pode demorar mais tempo do que o máximo esperado por um pedido HTTP.

Após o HTTP POST ser enviado ao SP, ao receber este POST o SP deve validar o conteúdo do mesmo para que possa concluir se estão presentes os parâmetros necessários à identificação do utilizador. Em caso afirmativo, deve extrair os mesmos da mensagem (HTTP POST) e obter o contexto do cliente através do Cid e devolver uma resposta HTTP com OK ao CHelper.

Ao receber o OK do SP, considerando que o Browser mantem o estado dos HTTP Redirect, o CHelper deve responder ao HTTP Redirect enviado pelo SP para que o Browser tenha conhecimento que a comunicação entre o SP e o CHelper terminou. Esta resposta é então processada pelo SP que inicia o processo de validação da assinatura. Se todo este processo ocorrer sem erros, e a informação envida pelo CHelper for validada pelo SP, então o utilizador encontra-se autenticado e o SP deverá ceder o recurso protegido requerido inicialmente pelo utilizador.

Pode-se por fim verificar que, com este sistema de comunicação entre CHelper e o SP, não existe a necessidade do servidor "web" estar preparado para autenticação com smartcards nem existe qualquer dependência dessa mesma autenticação nos navegadores uma vez que o CHelper que trata de todo o processo de acesso ao CC e o SP trata apenas do processo de autenticação do cliente.

Comprova-se com esta implementação que é viável o sistema de autenticação com recurso à utilização de "smartcards" como instrumento de autenticação, sem que para isso existam dependências ou diferenças na utilização de diferentes navegadores "web". Isto deve-se à natureza independente da aplicação criada para a resolução do problema apresentado neste trabalho.

Comprova-se também a não existência de uma dependência direta por parte do servidor "web" na manipulação com "smartcards", mas sim com certificados e pares de chaves. Como tal, esta implementação pode ser utilizada em vários sistemas de autenticação, sem existir a necessidade de uma alteração muito significativa aos mesmos.

Desta forma pensou-se na possibilidade de aplicar os resultados obtidos nesta secção a um sistema de autenticação utilizado em maior escala. Um sistema em que existam outras entidades além do utilizador e o fornecedor de serviços, bem como a possibilidade de existirem outros mecanismos de autenticação. Assim seria possível a utilizador optar por utilizar a autenticação com recurso ao CC ou uma outra disponibilizada pelo IdP.

Existem atualmente vários e diferentes sistemas de autenticação. Foi escolhido o sistema de autenticação federada SSO pela sua existência na Universidade de Aveiro e possibilidade de testes direto no mesmo com autorização da entidade gestora do mesmo. Este processo levou ao apresentado na secção 5.

5 Autenticação Federada SSO

A arquitetura apresentada na secção 4 e apresentada na Figura 4, mostrou ser um excelente ponto de partida para a autenticação "online" entre um cliente e um serviço recorrendo à utilização do CC. Como tal, apesar de não se encontrar no enquadramento inicial deste trabalho, pensou-se no que seria necessário para levar este tipo de arquitetura ainda mais longe e relacioná-la com outros mecanismos de autenticação existentes.

A secção 2.1.7 considerou cenários de "web" SSO, uma vez que é cada vez mais uma arquitetura utilizada para garantir a autenticação dos utilizadores num serviço centralizado e garantir também o acesso a recursos protegidos nos fornecedores de serviços de forma segura. Assim, evoluiu-se a arquitetura introduzida na secção 4, para que permitisse a integração de um sistema federado de SSO, integrado no IdP. Esta arquitetura é demonstrada na Figura 7.

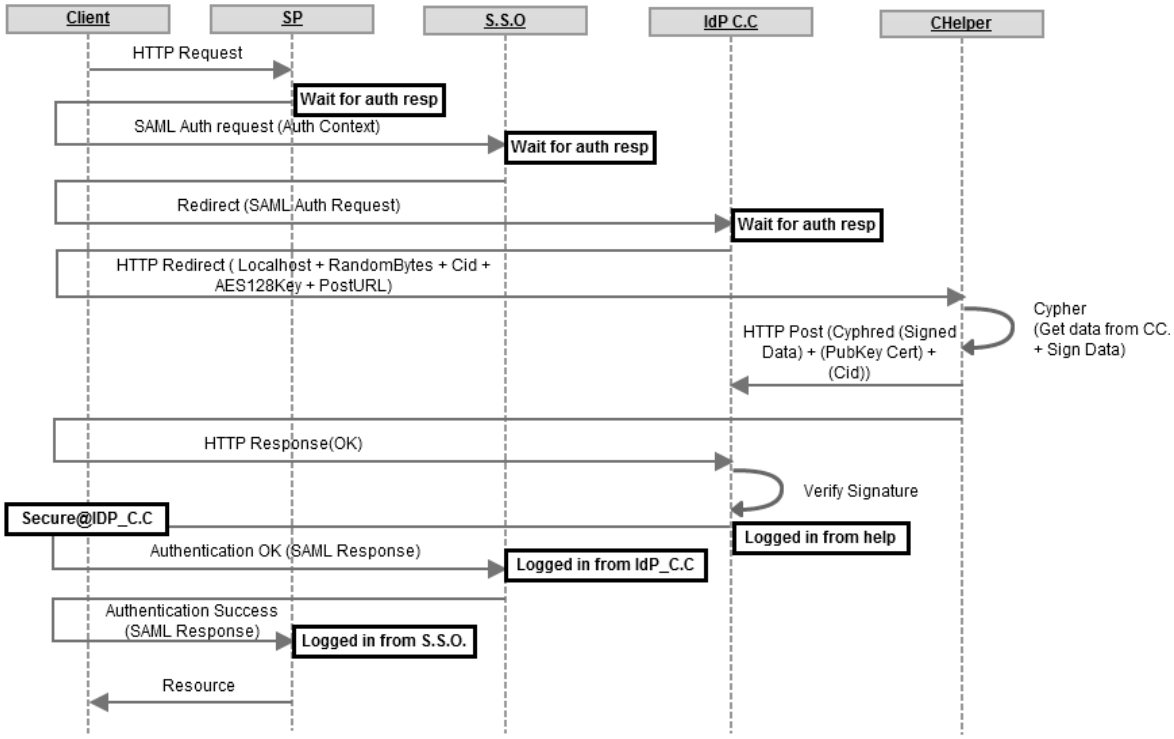

Figura 7 - Arquitetura SSO com recurso ao CHelper

Como podemos observar o número de atores desta arquitetura é maior do que numa normal arquitetura de SSO. Estes estão representados da seguinte forma:

- Cliente: No contexto deste trabalho, cliente é a designação atribuída a qualquer navegador utilizado para efetuar a comunicação entre os serviços, tanto no pedido

iniciado pelo utilizador, passando por todos os redireccionamentos necessários à troca de informação entre IdP, SP e SSO, até que a informação pretendida é finalmente obtida pelo utilizador.

- IdP_CC: O fornecedor de identidades é uma implementação de um servidor web que comunica com o CHelper, a fim de receber a informação de autenticação do utilizador. No fundo é um fornecedor de identidades um pouco diferente dos que estão disponíveis para uso online, pois terá de receber o pedido de autenticação do SSO em SAML e deverá contactar a aplicação CHelper através do processo descrito na secção 4. Depois de obter a informação pretendida, o IdP_CC, encaminhará devidamente a resposta SAML ao SSO. A arquitetura deste componente difere dos demais esquemas de SSO, pelo facto de receber o pedido de autenticação de um outro componente com capacidade IdP (neste trabalho, o SSO) e tratar de todo o processo de autenticação com a aplicação que corre na máquina do cliente, enviando no final de volta ao SSO a asserção com a resposta ao processo de autenticação. Fazendo com que o SSO contenha o informação de que o utilizador já se encontra autenticado até que este se desligue.
- CHelper é o IdP local definido na secção 4.
- SSO: É um servidor que efetua a agregação das identidades para o qual todos os clientes devem ser encaminhados, sempre que tentarem aceder a um recurso protegido. A função do SSO é a de permitir ao utilizador escolher um fornecedor de identidades adequado à informação que pretende aceder e ao método de autenticação que possui. Este servidor é criado com base num servidor de Shibolleth que reencaminhará os pedidos de autenticação SAML sempre que haja um pedido de acesso a um recurso protegido por um cliente que ainda não possua uma sessão válida. De acordo com a preferência deste, será encaminhado para o IdP correto a fim de se autenticar. Ao SSO cabe também a responsabilidade de, através de redireccionamento ao navegador do utilizador, fazer a entrega da resposta SAML com a autenticação validada ou não, dependendo de como correu o processo de autenticação no IdP.
- Service Provider: Como qualquer outro fornecedor de serviços, contém serviços que são requeridos pelos utilizadores, para o efeito deste trabalho são considerados quaisquer serviços disponibilizados pela Universidade de Aveiro como Service Providers. Para efeitos de teste foi autorizado pela Universidade de Aveiro o acesso ao número mecanográfico de um determinado utilizador, desde que este fosse devidamente autenticado com o CC.

O funcionamento inicial desta arquitetura é em tudo semelhante ao de um sistema de SSO normal. O Cliente efetua um pedido ao SP, que por sua vez verifica se o cliente já se encontra autenticado. Caso não se encontre, inicia o processo de autenticação enviando um SAML "Authentication Request" para o SSO. Como referido na lista acima, neste cenário o SSO age como um agregador de identidades permitindo ao cliente escolher o IdP pretendido para a sua autenticação. Se for escolhido um IdP normal, o pedido de "Authentication SAML" é reencaminhado para o respetivo IdP e o processo de autenticação segue conforme a norma desse IdP, não sendo acionada a arquitetura deste trabalho.

No entanto, caso seja selecionado a identificação com recurso ao CC, o pedido é reencaminhado para o IdP_CC. Ao receber este pedido, o IdP_CC inicia o processo descrito na secção 4, obtendo a informação necessária para autenticar o utilizador e, agindo como um agente de validação dessa mesma informação, devolve a mesma ao SSO, através de uma asserção SAML, a resposta ao pedido de autenticação. Este por sua vez, devolve ao SP a resposta ao pedido inicial de autenticação, fazendo com que o SP autorize ou não o acesso ao recurso protegido dependendo da resposta recebida. De notar que, para este processo ser possível, é imperativo que o SSO confie no IdP_CC pois é este que fará a validação dos dados recebidos do utilizador.

É importante pensar nesta arquitetura não só como um sistema de autenticação que pode funcionar com recurso ao CC, mas também como um complemento aos demais sistemas de autenticação existentes.

Recorrendo ao componente S.S.O. apresentado na Figura 7 é possível a um utilizador autenticar-se com um único método de autenticação e aceder a recursos específicos ou, se assim for pretendido, utilizar diferentes métodos de autenticação para acesso a diferentes recursos. Desta forma é possível atribuir diferentes níveis de segurança a diferentes recursos. Uma aplicação prática em que, a existência de diferentes tipos de autenticação seria importante, poderia ser uma consulta de dados que um aluno faz no portal da Universidade de Aveiro. Em que para visualização de informação seria apenas necessário o mecanismo de autenticação nome de utilizador e palavra-chave e alteração dos dados pessoais a autenticação com o CC.

Neste exemplo, o componente S.S.O. seria o responsável por gerir a autenticação do utilizador, informar o SP dos recursos que o utilizador teria disponíveis para aceder, mediante o tipo de autenticação efetuado, ou mesmo pedir um tipo diferente de autenticação ao utilizador, caso este pretendesse aceder a recursos aos quais não teria acesso com o método de autenticação utilizado.

Esta integração é importante, na medida em que vem contribuir com uma melhoria para os atuais sistemas de autenticação. Permitindo não só uma autenticação com os atuais métodos, mas também com recurso a “smartcards” sem necessidade de uma restruturação completa dos serviços existentes.

6 Implementação

As arquiteturas descritas nas secções 4 e 5 foram alvo de implementações e respetivos testes de funcionamento, a fim de ser possível provar que as mesmas arquiteturas são funcionais e trazem realmente um valor acrescido ao processo de autenticação com recurso ao CC.

A linguagem de programação usada durante o decorrer do trabalho foi maioritariamente Java, tanto para a criação da aplicação CHelper, como nas demais implementações de servidores Web e finalmente nas versões de teste da arquitetura. Foi utilizado C# apenas nos testes efetuados ao servidor Web IIS, durante o processo de teste à API deste servidor, uma vez que a programação para este servidor da Microsoft é forçosamente em C#.

A escolha de Java como linguagem de programação para este projeto fica a dever-se aos seguintes fatores:

- Multiplataforma.
- Fácil criação de um ficheiro único para iniciar e executar a aplicação.
- Linguagem ensinada atualmente na Universidade de Aveiro.
- Conhecimentos prévios na linguagem que permitiram agilizar a programação e respetivos testes.

A arquitetura deste trabalho contém um componente afeto ao equipamento do cliente/utilizador (CHelper) um novo componente externo não existente numa arquitetura SSO. A aplicação CHelper não tem qualquer tipo de instalação sendo a sua especificação e utilização demonstrada na secção seguinte (6.1).

6.1 Implementação do CHelper e protocolo de comunicação

Para a troca de mensagens de um serviço com o CHelper, no caso da secção 4, o SP sendo esse o serviço, a arquitetura presente para esta comunicação, Figura 4, obriga a uma troca de parâmetros entre o SP e o CHelper sempre que o SP pretende autenticar o utilizador. Estes parâmetros são os seguintes:

- RamdomByte: Corresponde a um conjunto de bytes aleatórios com 40bytes de tamanho, este parâmetro deve ser devolvido já assinado para que o SP possa comprovar o sucesso da assinatura com a chave privada do utilizador.
- Cid: Corresponde à identificação do cliente por parte do SP permitindo saber o contexto do cliente que está a efetuar a autenticação, este parâmetro deve ser devolvido inalterado para que o servidor possa manter o contexto da sessão do cliente.

- PostURL: Corresponde ao endereço para o qual o CHelper deve efetuar um HTTP Post, depois de assinar o parâmetro RandomBytes, este é enviado, assim como o certificado de chave pública do CC do utilizador obtido.
- AES128Key: Este parâmetro não é mais que uma chave AES com 128bits que é utilizada para cifrar o certificado de chave-pública e o RandomBytes já assinado, antes de serem devolvidos ao SP conferindo assim mais alguma segurança à informação do utilizador.

No envio do pedido de autenticação ao CHelper, uma vez que é o pedido é efetuado pelo SP existem alguns aspetos no início desta comunicação a ter em conta, sendo que o principal é como encontrar a localização do CHelper, presente no equipamento do cliente, e por conseguinte iniciar a comunicação com este.

Como já foi referido anteriormente na secção 4, o funcionamento básico de um qualquer servidor Web consiste em receber um pedido de um cliente e responder ao mesmo uma vez que o cliente iniciou a comunicação, por esta razão é irrelevante se o cliente se encontra por detrás de NAT ou uma "firewall", a resposta chegará ao cliente, no entanto isto não acontece se o servidor pretender iniciar ou aceder a um serviço que está a correr no cliente. Se como é o caso for o servidor a iniciar um serviço no cliente, este pedido pode ser barrado por uma firewall ou pode nunca chegar ao cliente devido ao NAT. Para contornar este problema, pensou-se que o ideal seria fazer um HTTP "Redirect" ao cliente (Browser), em que o destino deste era o "localhost". Ou seja, a própria máquina do cliente faz com que este inicie o serviço pretendido sem a preocupação do pedido nunca chegar ao cliente, uma vez que é o próprio Browser que se encontra na máquina do cliente a iniciar o serviço. Não sabendo inicialmente o resultado deste processo, o mesmo foi testado nos navegadores utilizados neste trabalho, IE, GC e Firefox, verificando-se o iniciar do servido no cliente com todos eles.

Ao receber o pedido de autenticação, o CHelper deverá analisar este verificando se contem os parâmetros necessários para efetuar uma assinatura com o CC e à criação da resposta para o SP, caso afirmativo inicia o processo de comunicação com o CC presente no mesmo equipamento de onde deve extrair o certificado de chave pública, neste processo a aplicação utiliza uma biblioteca, pteidpkcs11 [11], para iniciar o CC bem como fazer as verificações necessárias à validade do mesmo e uma outra, IAIK [22] para extrair e manipular o certificado de chave-pública.

Depois de assinar o parâmetro RandomBytes e o certificado de chave pública do utilizador com a chave AES enviada pelo servidor, o CHelper deve fazer um HTTP POST para o SP contendo os seguintes dados:

- Pubkeycert: O certificado de chave-pública do CC do utilizador.
- CyphredBytesCC: Corresponde ao conjunto de bytes recebido do SP já assinados com a chave privada presente no CC do utilizador.
- Cid: O mesmo recebido do SP para manter contexto da sessão do utilizador.

Existe também uma troca de informação entre o CHelper e o SP em que o CHelper envia um conjunto de informação ao SP através de um HTTP POST. O facto da comunicação entre o CHelper e o SP ser um HTTP POST, e não um HTTP Response ao HTTP Redirect inicial feito pelo SP, deve-se a dois fatores principais. Em primeiro lugar, um endereço (URL) com o tamanho necessário tem de ser suportado no navegador do utilizador. Em segundo lugar, o servidor ao qual o cliente pretende ligar-se tem de suportar o tamanho pretendido. Uma vez que a implementação do IdP_CC se enquadra no âmbito deste trabalho, não seria um problema escolher um servidor Web e prepara-lo para que permitisse um URL grande o suficiente. No entanto do lado dos navegadores não foi encontrado nenhum controlo deste tipo de parâmetros.

Apesar da norma HTTP/1.1 [20] não definir um tamanho máximo para um URL, este tamanho é limitado pelos navegadores em que diferentes navegadores permitem diferentes tamanhos máximos de URL [21]. A Tabela 4 mostra os navegadores verificados e a sua capacidade máxima no tamanho de URL.

Navegador	Tamanho máximo de URL suportado em bytes
Firefox	>8128
Opera	>8128
Google Chrome	>8128
Internet Explorer	2048

Tabela 4 - Tamanho máximo de URL suportado em vários navegadores

Os valores presentes na Tabela 4 foram testados no âmbito deste trabalho permitindo assim descobrir que o tamanho máximo que se deve utilizar para ter o máximo de compatibilidade com navegadores é de 2048. Tendo em conta que, devido ao aumento crescente no tamanho dos pares de chaves e, por conseguinte, o tamanho dos certificados de chave pública, não é possível garantir que este valor é suficiente no futuro, como tal optou-se pela troca de informação através de HTTP POST que não tem qualquer limite de restrição relativamente ao tamanho do conteúdo a enviar.

6.2 Experimentação

Para tirar partido das funcionalidades implementadas neste trabalho, é necessário iniciar um componente no computador/equipamento do utilizador, nomeadamente, a aplicação CHelper, pois a esta aplicação corresponde todo o processo de gestão do CC. É uma aplicação criada em Java, utilizando o IDE (Integrated development environment) NetBeans 7.0 com a versão de Java EE SDK 6 e JDK 6. Esta aplicação é de fácil utilização, não há necessidade de instalação uma vez que o Java permite criar um ficheiro executável java, pelo que é apenas necessário correr o mesmo. É Necessário que o equipamento que corra a aplicação suporte Java. Tanto no desenvolvimento, como na experimentação o equipamento utilizado tinha as seguintes características:

Processador: Intel Dual Core 2.0Ghz

Memória: 4G DDR 2 a 633Mhz

Placa Gráfica: Nvidia 9300GS

Depois do CHelper estar em funcionamento, basta colocar o CC no leitor de cartões (Figura 8) e ser-lhe-á pedido que coloque o PIN de identificação para que possa ser validada a presença do titular do referido cartão. Após este passo o utilizador está autenticado no fornecedor de identidades. Este fornecedor tem agora a responsabilidade de autenticar todos os pedidos efetuados pelo utilizador que se acabou de autenticar por tanto tempo quanto a sessão tenha disponível, ou até que o utilizador decida fazer "logout" da sessão.

Figura 8 - Leitor de smartcard com o CC inserido

Na Figura 9 é apresentada uma imagem onde se mostra o CHelper e as suas funcionalidades. É uma aplicação simples de utilizar pois foi criada a pensar em todos os utilizadores, mesmo os menos experimentes com a informática.

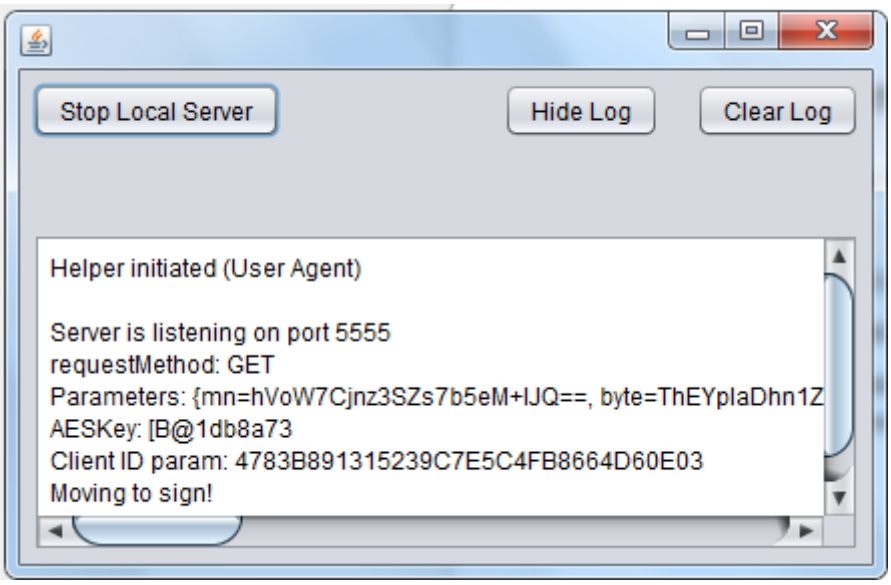

Figura 9 - Aplicação CHelper

O botão "Stop Local Server" é um botão com dupla funcionalidade, antes de iniciar o serviço apresenta o texto "Start Local Server" e permite iniciar o servidor de processamento de pedidos, depois de iniciado o servidor permite ao utilizador parar o mesmo. O botão "Hide Log" permite esconder as mensagens geradas no decorrer do processamento de pedidos. O botão "Clear Log" permite ao utilizador apagar todas as mensagens apresentadas na caixa de texto até ao momento.

Esta aplicação permite ao utilizador iniciar o servidor local para processamento de pedidos com o IdP_CC, como referido na secção 4, assim como esconder a janela de Log (Hide Log) e ainda apagar o Log. Para que um utilizador possa iniciar o programa é necessário ativar o servidor local (Activate Local Server/Stop Local Server). De notar que na Figura 9 o Log encontra-se em modo de depuração apresentando assim todos os dados e transações efetuadas durante o correr da aplicação. Numa versão final apenas as mensagens importantes seriam mostradas ao utilizador.

O acesso ao CC é efetuado através de uma biblioteca específica em Java (IAIK [22]) e a biblioteca fornecida pelos serviços do estado Português (pteidpkcs11 [11]). Para a utilização de um "smartcard" diferente do CC, é possível que exista a necessidade da utilização da biblioteca do referido "smartcard" para acesso e validação do mesmo.

6.3 Testes e verificação

Após o desenho da arquitetura e a sua implementação foram efetuados os respetivos testes e verificação de funcionamento da implementação do trabalho. Apesar de uma parte dos testes ter já sido efetuada no decorrer do trabalho a fim de se avaliar a necessidade do desenvolvimento da arquitetura de alguns componentes (secções 2.2.4 e 3), como foi o caso da arquitetura escolhida para o CHelper e respetivo protocolo de comunicação com um serviço externo (no caso da secção 4 o SP) em que para os mesmos se teve de estudar e testar antecipadamente o comportamento dos navegadores e servidores web. Durante o processo de implementação desta arquitetura foi necessário testar a comunicação entre os componentes de teste (CHelper e SP) para que se conseguisse concluir que todas as comunicações presentes no protocolo iriam atuar como o modelo teórico. Conseguiu-se chegar à conclusão que a comunicação é efetuada com sucesso e tanto o processo de HTTP "Redirect" como o HTTP POST que é sucedido, garantindo desta forma as trocas corretas de informação entre os componentes.

Foram também efetuados testes ao funcionamento básico das bibliotecas de acesso ao CC a fim de se obter resultados acerca da capacidade destas em retirar do mesmo o certificado de chave pública e respetiva assinatura com a chave privada ao que se verificou mais uma vez o sucesso das transações, conseguindo aceder com sucesso ao CC.

Por conseguinte, depois de devidamente encaixadas as peças referidas anteriormente, testou-se o funcionamento da arquitetura presente na secção 5 concluindo que o tempo de comunicação era ligeiramente superior, sendo que o mesmo foi contabilizado em aproximadamente 2 segundos a mais. Este facto fica a dever-se aos tempos de acesso ao CC por parte do equipamento de leitura do mesmo.

O processo desde a ligação inicial ao serviço até que seja obtida a informação pretendia é demonstrado nas figuras seguintes, Figura 10 e Figura 11.

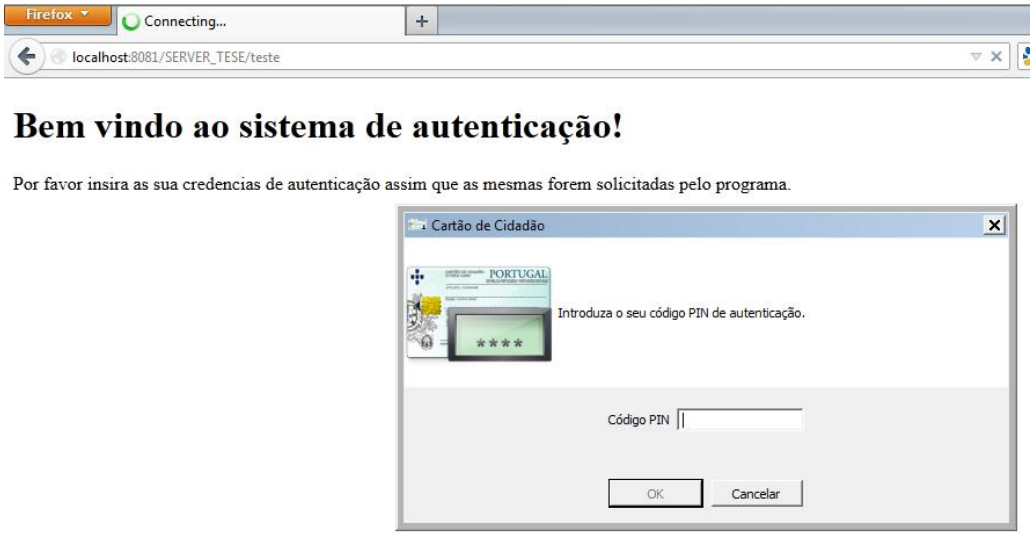

Figura 10 - Pagina inicial com pedido de autenticação

Todo o processo de troca de mensagens necessário a autenticação de um utilizador, é completamente transparente para o mesmo. Sendo que, no exemplo criado para este trabalho, ao ligar-se ao serviço para um pedido de informação, o utilizador é recebido com uma página de boas vindas. O "browser" é reencaminhado de seguida para uma página que carrega os módulos de acesso ao leitor de cartões e respetivo "smartcard", a fim de efetuar a leitura da informação presente no CC.

Após o utilizador ser autenticado com sucesso, é enviado um pedido ao serviço que retém os dados dos alunos na Universidade de Aveiro. Uma vez que este se trata de um exemplo em que é pretendido demonstrar a utilização deste sistema, não existe uma página intermédia que poderia pedir ao utilizar para selecionar o método de autenticação pretendido, antigo sistema de nome de utilizador e palavra-chave, ou o novo sistema de autenticação com o CC. Sendo que o processo assume de imediato que a autenticação pretendia é com recurso ao CC.

Este serviço devolve para efeitos de teste apenas o número mecanográfico e o nome de um aluno da Universidade como demonstrado na Figura 11.

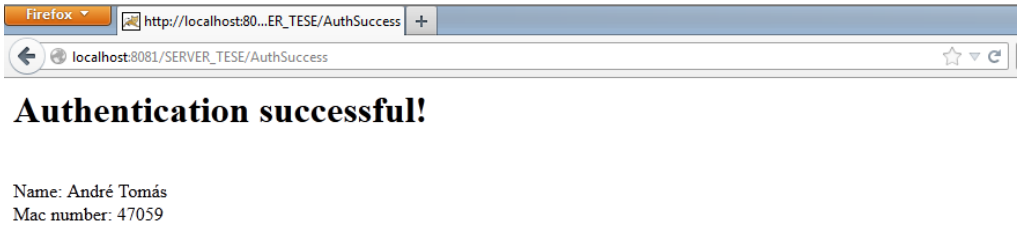

Figura 11 - Após a autenticação com sucesso

7 Conclusão e Trabalho Futuro

Este trabalho foi desenvolvido no âmbito da necessidade de um melhoramento na utilização de sessões seguras, com recurso a autenticação mútua e utilização de "smartcards" por parte dos utilizadores, tendo em conta o suporte oferecido pelos navegadores e servidores Web existentes.

Como foi possível perceber, existe uma grande incompatibilidade no que diz respeito à utilização de um “smartcard” num determinado navegador “web”. Foi também possível verificar que diferentes navegadores têm comportamentos diferentes na utilização do mesmo “smartcard”, sendo que por vezes o mesmo navegador tinha comportamentos diferentes no mesmo contexto. Demonstrou-se que não é possível depender das API’s fornecidas pelos servidores “web” para tratar sessões “web” seguras (HTTPS), visto que a grande maioria não está preparada para separar o contexto de sessão web (HTTP) do contexto de sessão segura (HTTPS).

Deste ponto se compreendeu a necessidade de um sistema que permitisse a não dependência de navegadores web para a comunicação com o CC, assim como a não dependência de servidores “web” para fazer a gestão da autenticação do utilizador.

Depois das pesquisas preliminares e início do trabalho verificou-se a viabilidade do mesmo, tornando-se possível criar um sistema que permita aos utilizadores aceder a diferentes recursos “online”, utilizando para isso diferentes meios de autenticação e permitindo assim oferecer diferentes níveis de segurança a diferentes serviços. Uma vez que o aumento nos níveis de segurança aumenta também a dificuldade de compreensão e utilização do serviço de autenticação, este trabalho vem melhorar a interação existente com serviços não críticos trazendo melhorias a facilidade de implementação nos sistemas de autenticação dos serviços críticos, garantindo sempre a segurança na autenticação e troca de informação.

Como foi possível reparar, o número de componentes presentes na arquitetura de SSO, secção 5, é consideravelmente maior do que o presente na arquitetura SP-CHelper, secção 4, visando melhorar a segurança e compatibilidade entre sistemas. Acabou por se aumentar os passos necessários, adicionando novas entidades. No entanto, e considerando que os demais sistemas de SSO são por natureza complexos, pode-se considerar que este pequeno aumento de complexidade pelo aparecimento de mais duas entidades, vem favorecer a compatibilidade entre sistemas e permitir a utilização do uso do CC por parte dos utilizadores como meio de autenticação independentemente do navegador Web que utilizem ou do tipo de servidor web existente.

No decorrer deste trabalho foi utilizado o CC como dispositivo para autenticação. Como referido anteriormente, o CC encontra-se na categoria de “smartcards” com uma estrutura PKI,

assim sendo, é de esperar que outros “smartcards” com estrutura PKI possam ser usados para autenticação num sistema deste género. Sendo que se existir alguma diferença é apenas necessário alterar o processo de acesso ao “smartcards” utilizado.

Para trabalho futuro, pode-se considerar uma implementação deste trabalho num sistema de autenticação já implementado com um elevado número de utilizadores como por exemplo na Universidade de Aveiro. Numa primeira fase, permitirá a utilização do CC como mecanismo de autenticação aos estudantes, uma vez, que atualmente apenas é possível aos mesmos efetuarem a sua autenticação através de nome de utilizador e palavra-chave. É importante fornecer diferentes mecanismos de autenticação aos utilizadores.

Numa fase final, deverão existir diferentes mecanismos de autenticação, em que os mecanismos mais básicos, como utilizador palavra-chave, darão acesso a informações mais básicas (e.g. visualizar dados), e mecanismos mais avançados, como “smartcards” oferecerão a possibilidade de manipulação de informação mais importante (e.g. inscrições em cadeiras e alteração de dados pessoais).

Numa visão mais futura deste trabalho, pode-se considerar a possibilidade de integração do mesmo com a maioria dos sistemas de autenticação existentes atualmente. Uma vez que não existe uma dependência de navegadores, qualquer utilizador estará apto a usar um serviço que implemente o conteúdo deste trabalho. Da mesma forma que, não existe uma necessidade de grandes alterações aos atuais servidores de autenticação, pelo que a integração deste trabalho com os mesmos, não deverá representar um desafio demasiado grande para restringir a sua aceitação. Contribuindo desta forma, para uma identificação mais segura em sessões “web”.

Bibliografia

[1] M. Burrows, M. Abadi, R. Needham. Authentication: A Practical Study in Belief and Action. TARK '88 Proceedings of the 2nd conference on Theoretical aspects of reasoning about knowledge, San Francisco, CA, 1988.

[2] N. Ferguson, B. Schneier. Practical Cryptography. John Wiley and Sons. ISBN 0-471-22357-3. April, 2003.

[3] W. Diffie, M. E. Hellman. New Directions in Cryptography. IEEE Transactions on Information Theory, vol. IT-22, pp. 644–654, November 1976.

[4] S. Cantor, J. Kemp, R. Philpott, and E. Maler. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard saml-core-2.0-os, March 15, 2005.

[5] B. Fitzpatrick, D. Recordon, D. Hardt, J. Hoyt. Openid authentication 2.0 - Final, December 2007. http://openid.net/developers/specs/.

[6] A. K. Lenstra, B. Dodson, J. Hughes, P. Leyland. Factoring Estimates for a 1024-bit RSA Modulus. ASIACRYPT 2003: 55-74. April 28, 2003.

[7] M. Winslett, N. Ching, V. Jones, I. Slepchin. Using digital credentials on the World Wide Web, Journal of Computer Security. IOS Press, 1997

[8] N. K. Ratha, J. H. Connell, R. M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. IBM systems Journal, vol. 40, pp. 614–634, 2001.

[9] S. Tarkoma, J. Heikkinen. From End-to-End to Trust-to-Trust. Seminar on Network Security. Helsinki University of Technology - Department of Computer Science and Engineering, Autumn 2008.

[10] Cartão de Cidadão – Portal do Cidadão. Retirado de http://www.cartaodecidadao.pt/ a 25/09/2012.

[11] Manual técnico do Cartão de Cidadão para a versão 1.24.1 (Windows) e 1.23.1 (Mac e Linux) retirado de https://www.portaldocidadao.pt/ccsoftware/CC%20Technical%20Reference%20v1.24.1%20PT.pdf em 27/09/2012.

[12] C. Adams, S. Lloyd. Understanding PKI: concepts, standards, and deployment considerations. Addison-Wesley Professional. ISBN: 0-672-32391-5, 2003.

[13] Mike Hendry. Multi-application Smart Cards: Technology and Applications. Cambridge University Press. July 23, 2007.

[14]R. Wang, S. Chen, X. Wang. Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services. In Proceedings of the 33rd IEEE Symposium on Security and Privacy (IEEE S&P), May 2012.

[15]Ben Adida. Sessionlock: securing web sessions against eavesdropping. WWW '08 Proceedings of the 17th international conference on World Wide Web, 2008.

[16]R. Oppliger, R. Hauser, D. Basin. SSL/TLS Session-Aware User Authentication - Or How to Effectively Thwart the Man-in-the-Middle. Computer Communications, 29(12):2238–2246, August 2006.

[17]A. Hess, J. Jacobson, H. Mills, R. Wamsley, K.E. Seamons, B. Smith. Advanced Client/Server Authentication in TLS. Retirado de: HTTP://uran.donetsk.ua/~masters/2010/fknt/chernobay/library/article4.htm a 03/06/2012.

[18] N. Ragouzis, J. Hughes, R. Philpott, Eve Maler. Security Assertion Markup Language (SAML) V2.0 Technical Overview. Working Draft 10, October 2006

[19]Apache Tomcat 7 - SSL Configuration HOW-TO. Retirado de HTTP://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html a 05/10/2012

[20]R. Fielding, UC Irvine, J. Gettys, Compaq/W3C, J. Mogul, Compaq, H. Frystyk, W3C/MIT, L. Masinter, Xerox, P. Leach, Microsoft, T. Berners-Lee. RFC 2616, Hypertext Transfer Protocol - HTTP/1.1. HTTP://www.w3.org/Protocols/rfc2616/rfc2616.html, June 1999.

[21]What is the maximum length of a URL? Retirado de HTTP://stackoverflow.com/questions/417142/what-is-the-maximum-length-of-a-url a 12/10/2012.

[22]IAKI-JCE retirado de http://jce.iaik.tugraz.at/sic/Media/product_pdfs/IAIK-JCE a 30/03/2012.

[23]R. Housley, SPYRUS, W. Ford, VeriSign, W. Polk, NIST, D. Solo, Citicorp. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. http://www.ietf.org/rfc/rfc2459.txt, January 1999.

[24]Mozilla Firefox - retirado de http://www.mozilla.org/en-US/firefox/new/ a 10/05/2012.

[25]Internet Explorer - retirado de http://windows.microsoft.com/en-US/internet-explorer/download-ie a 10/05/2012.

[26]Google Chrome - retirado de https://www.google.com/intl/en/chrome/browser/ a 10/05/2012.

[27]Opera Browser - retirado de http://www.opera.com/ a 10/05/2012.

[28]Tomcat - Open source software implementation of the Java Servlet and JavaServer Pages technologies. Retirado de http://tomcat.apache.org/index.html a 16/11/2011.

[29]IIS - Internet Information Services (IIS) for Windows. Retirado de http://www.iis.net/ a 19/01/2012.

[30]Apache - The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. Retirado de http://httpd.apache.org/ a 13/02/2012.

Anexo I

Para uma melhor compreensão da comunicação entre os componentes deste trabalho é apresentado de seguida, neste anexo, um diagrama de blocos. Este diagrama contém os componentes agrupados na forma como deveram ser utilizados:

Client – Contem o serviço CHelper e o respetivo navegador do utilizador.

S.S.O and IdP_CC web services – Contém o componente S.S.O para a gestão e verificação de autenticação e o IdP_CC para novas autenticações com CC.